RICE UNIVERSITY

Improvement of Image Reconstruction Speed with GPU in Cone Beam CT Breast Imaging

by

Shuaiping Ge

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

APPROVED, THESIS COMMITTEE:

Rebecca Richards-Kortum, Chair
Professor Bioengineering

Chris C. Shaw, thesis advisor
MD Anderson Cancer Center, U. of Texax

Jianpeng Ma,
Professor Bioengineering

Huey W.Huang
Professor Physics & Astronamy

Houston, Texas
December, 2009

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

Improvement of Image Reconstruction Speed with GPU in Cone Beam CT Breast Imaging

by

Shuaiping Ge

Breast cancer is the most common and the second lethal cancer among women in the United States. Our group is constructing a dedicated cone beam breast CT (CBCT) system to provide true 3D image to improve the screening and diagnostic of breast cancer. Our result shows that dedicated CBCT out-perform a lot than conventional CT when detecting micro-calcification which is essential to the detection of early stage breast cancer. I also explored the possibility of using the super parallel computing power of GPU with CUDA environment to deal with data-immense and computationally-intensive image reconstruction process of CBCT. My results show that FDK algorithm image reconstruction with GPU is over 10 times faster than that with our PC cluster system. The faster and accurate image reconstruction implies potential new applications in diagnostic and therapy technology.

Acknowledgements

I would like to express my sincere gratitude towards my advisor Dr. Chris C.Shaw and our collaborators for their invaluable guidance and support given to me during the course of this research. In particular, I would like to thank Dr.Chao-Jen lai, Dr.Tinsu Pan, Dr.Lingyun Chen and Ms.Tao Han for their help and invaluable suggestions.

Contents

Chapter 1
Background Concept

Breast cancer is the most common and the second leading death cancer among women in the United States. The chance of developing invasive breast cancer at some time in a woman's life is about 1 in 8 (12%). And the chance that breast cancer will be responsible for a woman's death is about 1 in 35 (about 3%). Early detection of breast cancer is critical for patient survival. Some basic ideas of breast cancer and breast imaging modalities are introduced in this chapter.

1.1 Breast Cancer

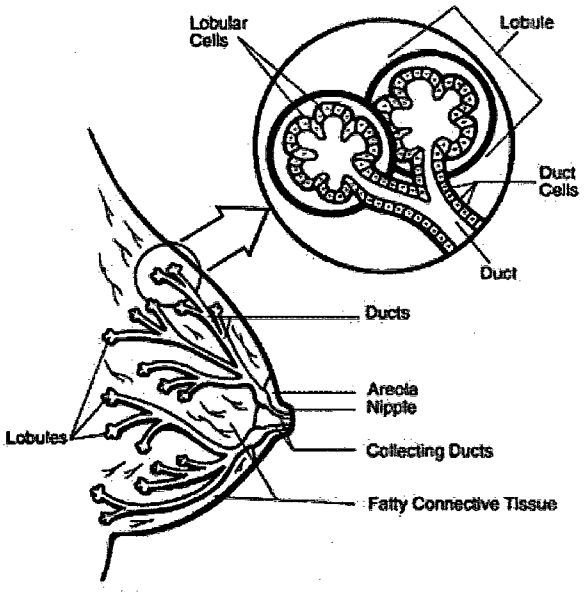


Figure 1-1. The structure of normal breast[1].

A normal female breast is made up mainly of lobules (milk-producing glands), ducts (tiny tubes that carry the milk from the lobules to the nipple), and stroma (fatty tissue and connective tissue, blood and lymphatic vessels), as showed in Figure 1-1.

Breast cancer is a malignant tumor that origins from breast. There are four types of common breast cancers:

Lobular carcinoma *in situ* (LCIS) (*in situ* means that the cancer cells remain confined to where it origins, not invades into other tissue or organs), begins in the lobule and does not grow outside the lobules. It is not a true cancer (sometimes classified as non-invasive breast cancer), but women with LCIS do have a higher risk of developing an invasive breast cancer.

Ductal carcinoma *in situ* (DCIS), means cancer cells are grown inside the ducts but have not spread into the surrounding breast tissue. DCIS is the most common type of non-invasive breast cancer.

Invasive lobular carcinoma (ILC), starts in the milk-producing glands (lobules). ILC can spread (metastasize) to other parts of the body through the lymphatic system and bloodstream. About 1 out of 10 invasive breast cancers are ILCs.

Invasive ductal carcinoma (IDC), starts in a duct of the breast, breaks through the wall of the duct, and grows into the fatty tissue of the breast. IDC may be able to spread (metastasize) to other parts of the body through the lymphatic system and bloodstream. IDC is the most common type of breast cancer, about 8 of 10 invasive breast cancers are IDC.

There are two kind of indicators when screening breast cancer: calcification and mass.

Calcifications are tiny mineral deposits within the breast tissue. There are two types of calcifications: Macro-calcifications are coarse (large) deposits that are often caused by aging of the breast arteries, old injuries, inflammation or benign fibrocystic change, which are non-cancerous conditions. Micro-calcifications are tiny (less than 1/50 inch) specks of calcium in the breast. They may appear alone or in cluster. Micro-calcification deposits mean higher possibility of cancer presenting. The shape and layout of micro-calcifications help the radiologist in evaluating the likelihood of cancer. About half of the cancers detected by mammography appear as a cluster of micro-calcifications. And almost 90% of cases of ductal carcinoma in situ are associated with micro-calcifications.

Masses may occur with or without associated calcifications. A mass is any group of cells clustered together more densely than the surrounding tissue. A cyst (a non-cancerous collection of fluid in the breast) may appear as a mass on x-ray images. Breast ultrasound or aspiration with a needle is required to confirm whether it is a cyst. A mass (not cyst) with calcifications can be caused by benign breast conditions or by breast cancer. The size, shape, and margins (edges) of the mass help the radiologist judge how likely it is a cancer tumor.

Early detection of breast cancer is critical to patient survival, table 1-1 shows 5-year breast cancer survival rate by stage. After seven years, the survival rate decrease for each stage.

Table 1-1. 5-year breast cancer survival rate by state

Stage	0	I	IIA	IIB	IIIA	IIIB	IV
Survival rate	100%	100%	92%	81%	67%	54%	20%

Source: American Cancer society.

1.2 Current Breast Imaging Modalities

1.2 .1 Mammography

Mammography is a low energy, low dose x-ray examination of the breast, which produces high resolution and high contrast x-ray image of inner breast tissue. Mammography is currently the only exam approved by the U.S. Food and Drug Administration (FDA) to screen for breast cancer in women who do not show any signs or symptoms of the disease. During mammography, each breast will be positioned on a platform (with a special film cassette under it) and compressed with a paddle. The x-rays pass through the breast, loss part of the energy due to the absorption by tissue, expose the film. Screening mammography takes x-rays from two views for each breast. One is top to bottom and the other is a side view. Diagnostic mammography will take more views of breast. Mammography can detect approximately 85% of breast cancers.

However, mammography is a two dimensional projection imaging technique. There are several drawbacks associated with this projection imaging technique. Since the detection of the lesion is determined not only by the difference of linear attenuation coefficient between the lesion and the background tissue but also by tumor size and background thickness and structures, Detection and visualization of lesions could be either limited by the contrast signal-to-noise ratio or obscured by the overlapping background structures. Tumor and micro-calcifications can be obscured by the overlapping of tissue structure.

1.2 .2 Ultrasound

Breast ultrasound, also known as sonography or ultrasonography, is frequently used to evaluate breast abnormalities. Breast ultrasound use high-frequency sound waves to obtain breast tissue image. Ultrasound device records and displays sound wave which is produces by the device and reflected by the edge of the tissue in real time. Ultrasound is excellent at imaging cyst (fluid-filled pockets inside the breast). Ultrasound can quickly determine if a suspicious area is in fact a cyst or a dense mass (which may require a biopsy to determine if it is a cancer tumor). The absence of radiation makes ultrasound ideal for studying breast abnormalities in women who are pregnant. Ultrasound has excellent contrast resolution. But its spatial resolution is very bad (due to the characteristic of sound wave), it cannot provide as much detail structures as a mammogram can do. Micro-calcifications which are often the first indication of breast cancer is also undetectable for ultrasound images.

1.2 .3 MRI

Magnetic Resonance Imaging, or MRI, uses radio waves and a strong magnetic field to produce two or three-dimensional images of internal body structure. In the strong magnetic field, the magnetic moments of hydrogen nuclei (or protons, contained by water in the body) align with the field. During the examination, a radio signal is turned on, which cause the protons to alter their alignment relative to the direction of the field. Turning off the radio signal cause the protons to return to original magnetization alignment and release some energy which can be detected by the scanner. Proton's resonate frequency depends on the strength of the magnetic field. Turning on gradients

coils (producing a magnetic field gradient) helps to get the position information of protons in the body.

MRI has been approved by FDA for use as a supplemental tool, in addition to mammography, to help diagnose breast cancer. MRI provides greater contrast between different soft tissues of the body than computed tomography (CT) or mammography does. MRI is a painless procedure that can image and clearly map out soft tissues, producing images that are extremely precise without exposing patients to radiation or radioactive solution. MRI is excellent at imaging the augmented breast and staging breast cancer. The technique is also believed to have significant potential for distinguishing between benign and malignant lesions. Compared to mammography, MRI exam has the advantage to allow breast images to be taken in any plane and from any orientation.

There are also limitations associated with MRI. First, MRI cannot always distinguish between cancerous and non-cancerous abnormalities, which lead to too many positive fault and unnecessary breast biopsies. Another drawback of MRI is that it is unable to image tiny specks of calcium which indicate breast cancer. While with x-rays, calcifications can be indicator of early-stage breast cancers such as ductal carcinoma in situ (DCIS). An additional disadvantage of MRI is that it costs about 10 times than that of x-ray mammography.

Chapter 2
Comparison of Dedicated Cone-beam Breast CT with Conventional CT for Detection of Micro-calcification

Due to its high spatial resolution, good contrast of image quality, mammography is the only FDA approved screen exam modality for breast cancer in women who do not show any signs or symptoms of the disease. But mammography is a two dimensional projection image, overlapping tissue structures in mammography obscure and prevent some lesions and micro-calcifications to be detected. Computed tomography (CT) of the breast can effectively overcome the disadvantage and provide 3-D representation of breast anatomy. However, conventional CT scanners have been developed to scan around the chest rather than the breast itself, leading to substantial radiation dose inefficiency and degrading spatial resolution, tissue contrast of image. Dedicated 3-D breast cone-beam CT (CBCT) has been proposed and investigated in recent years. Our group is focused on developing dedicated CBCT system.

2.1 Principle of CT

Computed tomography (CT) technology reconstructs x-ray attenuation values of each small volume of tissues from x-ray projection images. When a narrow beam of monoenergetic photons with energy E_0 and intensity I_0 passes through a homogeneous absorber of thickness x can be expressed as:

$$I = I_o e^{-\mu(\rho,Z,E_o)x} \tag{2.1}$$

Where μ, ρ, Z are the linear attenuation coefficient, density of the absorber, and atomic number, respectively. For human tissues, the linear attenuation character μ is not homogeneous, $\mu(\rho, Z)$ is a space-variant function $\mu(x, y, z)$ depending on the distribution of the tissue structure. By delivering a monochromatic x-ray with photon intensity I_o in the t path, the transmitted x-ray intensity is:

$$I = I_o e^{-\int \mu(t)t} \tag{2.2}$$

Taking the logarithm and rearrange (2.2), one can obtain projection data P(x):

$$P(x) = -\ln\left(\frac{I}{I_o}\right)$$

$$= \int \mu(t)t \tag{2.3}$$

Here *P(x)* is equivalent to a simple integration or summation of the total attenuation coefficients along the x-ray path. In the real patient case, suppose a slab of tissue is irradiated, we can divide the slab into voxels of width Δx and assume each voxel has its own attenuation coefficient $\mu_i(x,y)$, see Figure 2-1. Equation (2.3) in a digital form becomes:

$$P(x) = \sum_{i=1}^{N} \mu_i(x, y)\, \Delta x \tag{2.4}$$

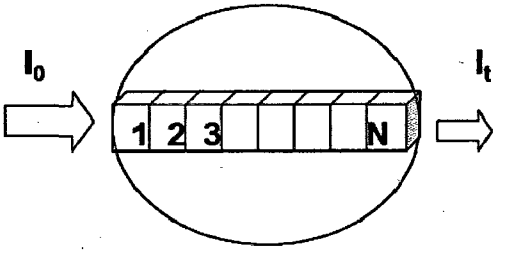


Figure 2-1. Illustration of material attenuation for x-ray beam. Any nonuniform object can be subdivided into multiple elements. Within each element, a uniform attenuation coefficient can be assumed.

Obviously each pixel of the projection image represents the summation or line integral of the attenuation coefficients of an object along a particular ray path. CT image reconstruction technology utilizes projection images from different angles to estimate the attenuation distribution of the scanned object through special algorithms.
Depend on the collimation shape of the x-ray beam; three different types of CT are identified: parallel beam, fan beam and cone beam CT.

2.1.1 Parallel Beam CT

In parallel beam CT, a pencil shaped x-ray beam is delivered to transmit through the object. A set of projection images is taken along x-ray paths that are parallel to each other and are uniformly spaced, as shown by the green lines in Figure 2-2. This measurements form a projection of view. Then the x-ray beam turns a small angle; and the same measurement process is repeated as shown by the purple lines in Figure 2-2. Keeping the angular increment constant and the scanned object stationary, The measurement process continues to cover the entire 360° (theoretically, only 180 ° of parallel projection are necessary)

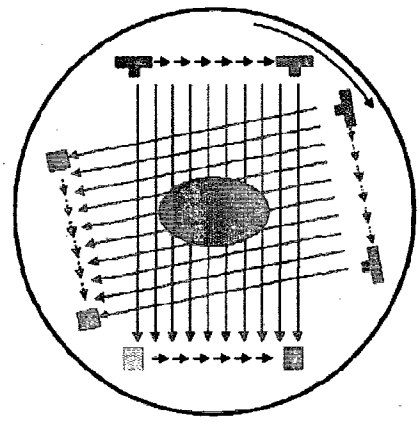


Figure 2-2. Illustration of parallel beam CT.

2.1.2 Fan Beam CT

In parallel beam CT, the source and the detector are translated for each rotation angle. This leads to long scanning times, which is particularly unwanted in medical diagnostics. Fan beam CT offers solutions to overcome the disadvantage. In fan beam CT, the x-rays are finely collimated resulting in a divergent fan beam x-rays which are detected by a linear x-ray detector. For later generation fan beam CTs, the fan angle of the beam is widened, allowing the beam to cover the entire object. A stationary circular ring detector array is built in the machine, and the scanner is rotate-only, no translational movement is required, as show in Figure 2-3.

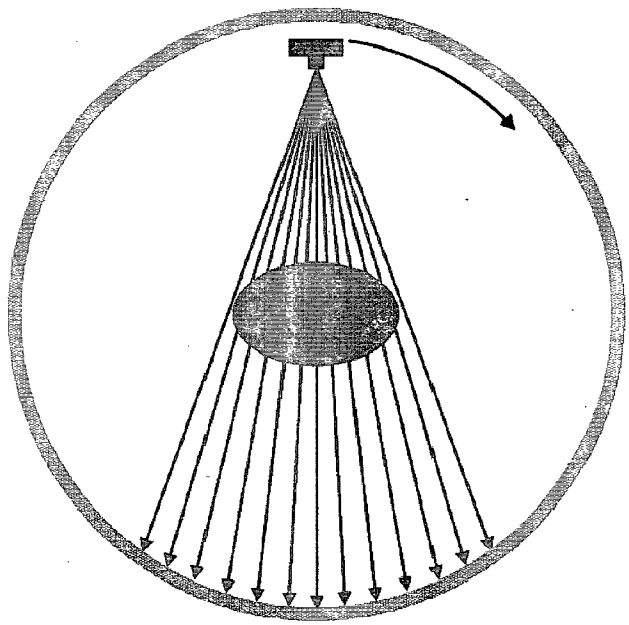


Figure 2-3. Illustration of fan beam CT.

2.1.3 Cone Beam CT

Cone beam CT uses a 2D detector to obtain a 2D array of projection image, as show in Figure 2-4. Cone beam X-rays penetrate the whole target, forming a complete 2D

projection imaging. All slices of projection data in one angle is acquired after just one shooting. For the parallel beam CT and fan beam CT, one slice of projection image is taken after a 360° rotation, and then the generator and detector (or patient) is moved to get the next slice of projection image. For cone beam CT, usually only one 360° gantry rotation is needed to acquire whole slices of projection images. The advantage of cone beam x-ray CT is that it has short examination time and saves x-ray tubes. The short scan time reduces the probability of patient motion and allows more isotropic spatial resolution. And usually, cone-beam CT is cheap than conventional CT. The main disadvantage of the cone beam CT is the difficulty of avoiding large amounts scattered radiation detection. The scatter-to-primary rations are about 0.01 for single-ray CT, 0.05-0.15 for fan-beam CT and 0.42-2 for cone beam CT.

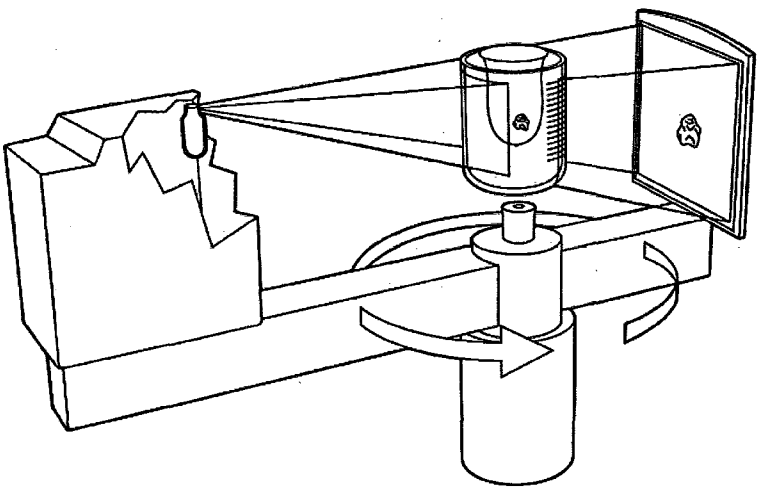


Figure 2-4. Illustration of cone-beam CT. Cone beam X-rays penetrate the whole target, forming a complete 2D projection imaging.

2.2 Cone Beam Breast CT

Our group constructed a flat-panel based cone beam CT system for dedicated breast imaging. The special geometry of our cone beam CT system makes only breast exposure to radiation, which saves dose and make it possible to provide true 3D breast image with radiation dose comparable to two-view mammography.

For conventional CT scanner, the images were acquired transversely and the x-ray beams penetrate the thoracic cavity. With this geometry, chest beneath breast is exposed to radiation, leading to unnecessary tissue exposure and degrading image resolution. Also, cardiac and respiratory motions have the potential to reduce image quality.

A dedicated cone beam breast CT system usually consists of an x-ray source and FP detector oppositely placed underneath a table on which the patient would lie in the prone position with one breast drawn downward through an opening, as show in Figure 2-5. X-ray tube and detector rotate around and scan the breast beneath the table in the coronal plane, as show in Figure 2-4. Half cone scan is preferred in order to prevent the exposure of the tissue in the thoracic cavity. As in mammography, the tube is specially designed to have the focal spot as close to the table or patient's chest as possible. In order to get information of tissues as close to the chest as possible, the detector should also be specially designed to let the side of the active area as close to the patient's chest as possible.

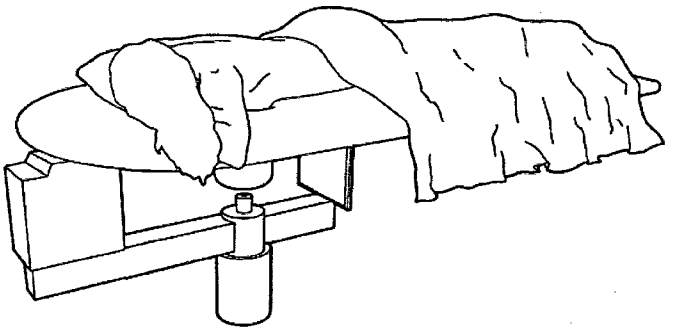


Figure 2-5. Illustration of dedicated breast CT system. A patient lie on a table in the prone position with one breast drawn downward through an opening, X-ray tube and detector rotate around and scan the breast beneath the table in the coronal plane

2.3 Comparison of Dedicated Cone-beam Breast CT with Conventional CT for Detection of Micro-calcification

Detection and visualization of micro-calcification are essential to the detection of breast cancer especially early stage breast cancer, which is critical to improving survival and decreasing mortality of breast cancer patients. Dedicated cone-beam breast CT (CBCT, shown as in Figure 2-4) has been proposed and investigated in recent years[2-5].

2.3.1 Materials and Methods

The pendant geometry of CBCT system allows only the breast to be scanned and imaged. It may help detect and visualize small micro-calcifications. My investigation compared the visibility of micro-calcifications in CBCT image with that in conventional CT image. Calcium Carbonate grains of various size ranges (180-200μm, 200-212μm, 212-224μm, 224-250μm and 250-280μm) were used to simulate micro-calcifications. Micro-calcifications from the same group were arranged to form a 5x5 cluster which was imbedded in a small paraffin rod. Two bowl shaped plastic containers filled with paraffin

were used to simulate breasts (Figure 2-6. A, B). The chest sides of the two phantoms were molded with proper contours to fit the chest of an anthropomorphic chest phantom so that they could be integrated to mimic a female patient in the prone position with two

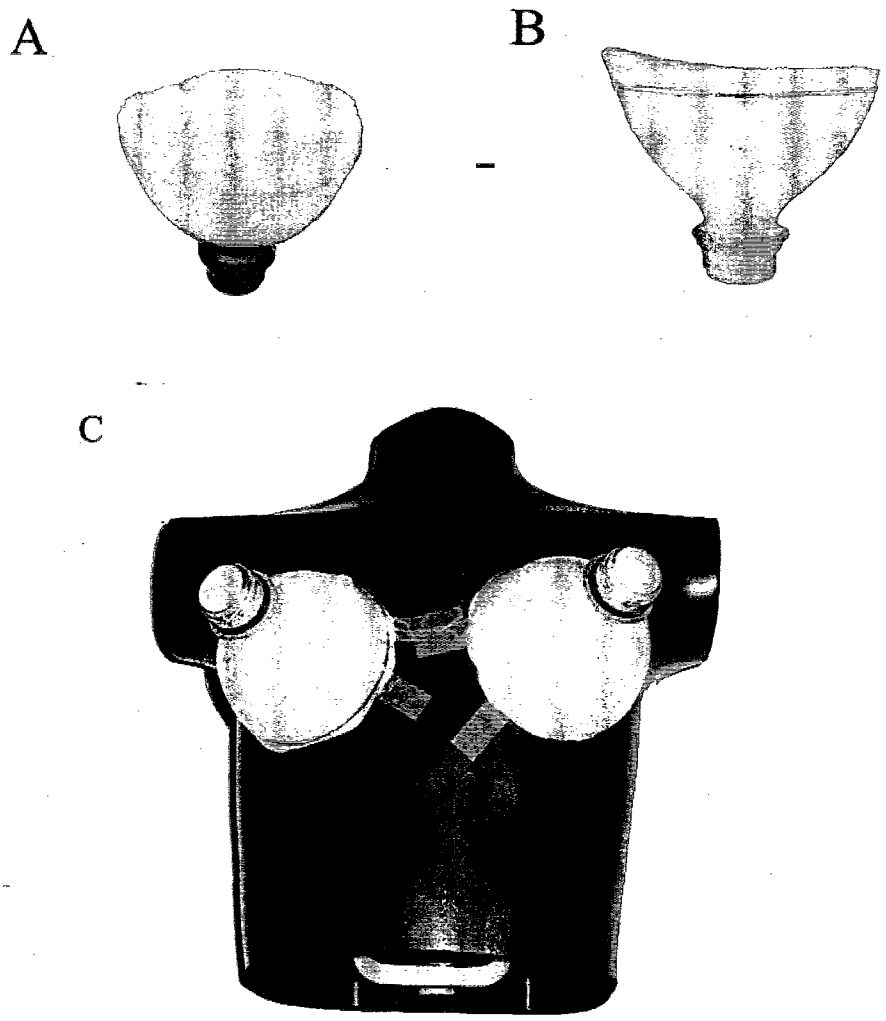


Figure 2-6. Illustration of chest phantom. (A) Image of breast phantom A, three holes are drilled through the coronal plane. (B) Image of breast phantom B, four holes are drilled through the axial plane (lowest one is useless). (C) Image of an anthropomorphic chest phantom with the two breast phantom attached.

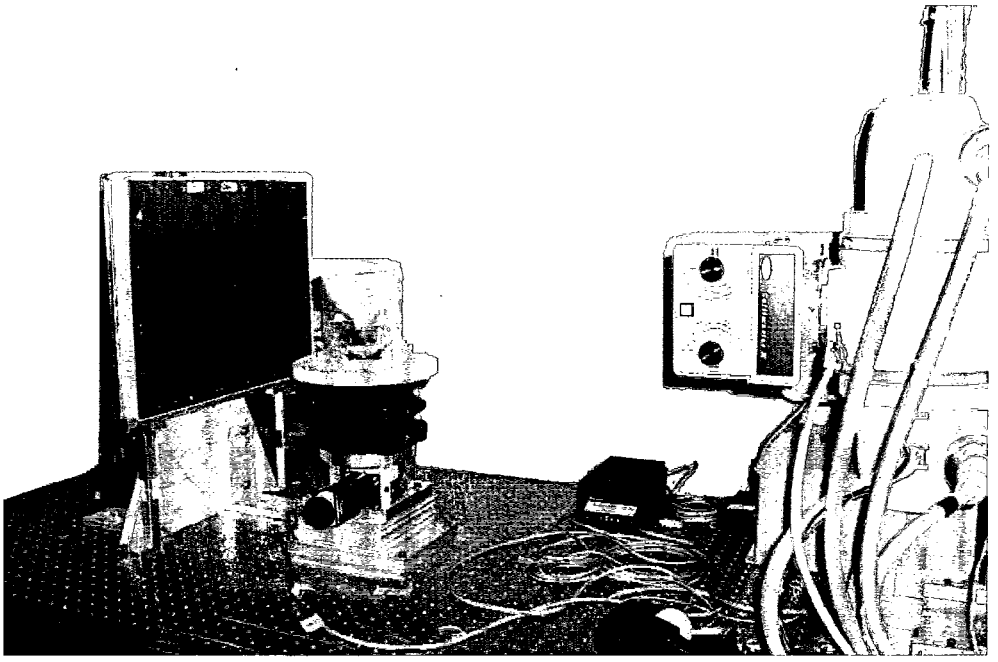


Figure 2-7. A stationary gantry, rotating phantom system was used to conduct the CBCT imaging experiments. The system consists of tungsten target x-ray tube (G-1592, Varian Medical Systems), a a-Si:H/CsI flat panel (FP) based detector (Paxscan 4030CB, Varian Medical Systems) and a step motor driven rotating table.

breasts protruding downward (Figure 2-6. C). Three 1.6cm diameter holes were drilled in the breast phantoms to allow the paraffin rods with imbedded calcifications to be inserted for imaging. One hole is located at the center of the breast phantom, the other two are located 2.5cm and 3.5cm from the center on the two opposite side.

The phantoms were scanned as follow.

1. Breast phantom A was scanned with a bench-top experimental CBCT system (Figure 2-7), configured for breast imaging. The system consists of a conventional tungsten target x-ray tube (G-1592, Varian Medical Systems) and an amorphous silicon/cesium iodide (a-Si:H/CsI) flat panel (FP) based detector (Paxscan 4030CB, Varian Medical Systems). The focal spot size used was 0.6mm and the

pixel size of the flat panel is 194 μm. The source to image and source to iso-center distances are 100 and 75 cm respectively, corresponding to a magnification factor of 1.33 at the iso-center. In the CBCT system, 300 projection images over 360° were acquired at 80kvp, 13mA and 80kvp, 27mA. The projection images were reconstructed with the Feldkamp algorithm[6].

2. Breast phantom A was scanned with a conventional multi-detector CT system (GE Discovery STE PET/CT Scanner). The slice thickness was 0.65mm. The x-ray tube was operated at 120kvp or 80kvp with 200mAs, 100mAs and 50mAs per revolution. Bone filter was used for image reconstruction.
3. Chest phantom C (with the two breast phantoms attached) was scanned by the same conventional CT. The x-ray tube was operated at 120kvp, 400mAs, or 80kvp, 400mAs. Bone filter was used for image reconstruction.
4. In each of the scanning proceedings, the three paraffin rods embedded with different sizes of calcifications were alternately inserted at different location to study the position effect, as show in Figure 2-8.

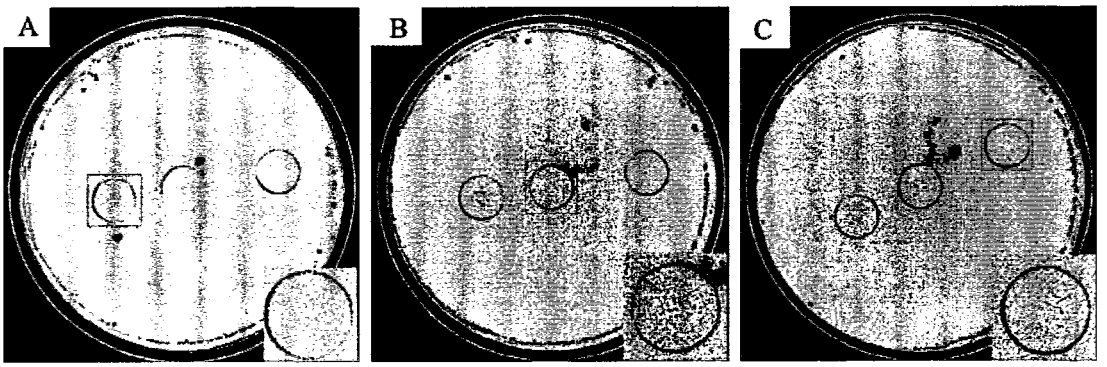


Figure 2-8. Image of 212-224μm micro-calcifications alternately inserted into holes located at 2.5cm, 0cm, and 3.5cm from the center for scan to study the position effect.

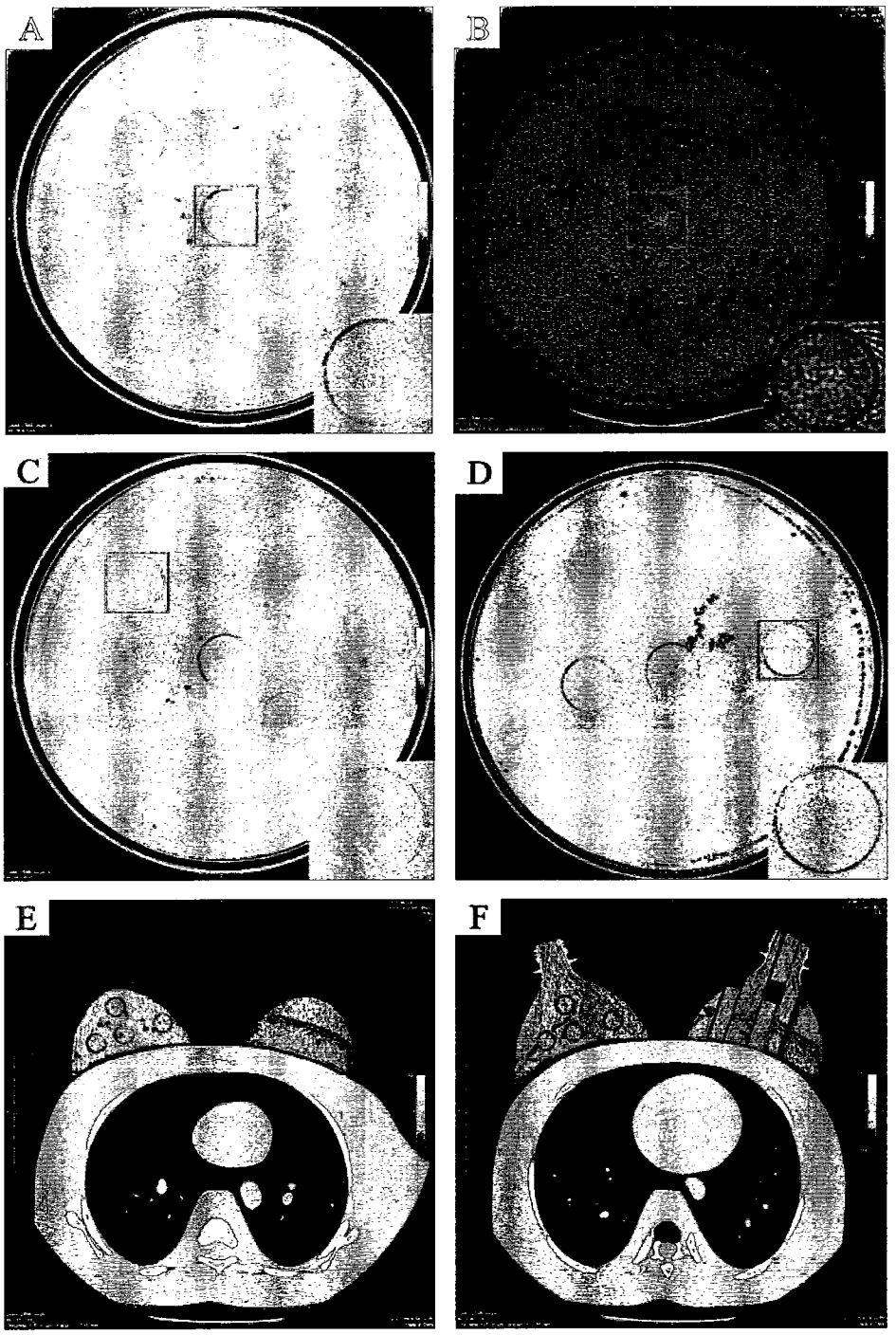


Figure 2-9. Comparing visibility of calcifications in CBCT images (80kvp, 27mA) with that in conventional CT images. (A) CBCT image of 212-224μm calcifications. (B) Conventional CT image of 212-224μm calcifications (80kvp, 200mAs). (C) CBCT image of 200-212μm calcification. (D) CBCT image of 180-200μm calcification. (E, F) Conventional CT images of chest phantom C (80kvp, 400mAs).

2.3.2 Results

The visibility of the calcifications in CBCT images was found to be significantly better than that in the conventional CT images. Figure 2-9. compares the visibility of 212-224μm calcifications scanned with CBCT system (Figure5 A) and with conventional multi-detector CT system (Figure5 B) when only breast phantom A was scanned. Due to slight misalignment, the micro-calcifications were located on several consecutive slices with each slice contains only part of the visible micro-calcifications. Calcifications as small as 180-200μm were visible in the CBCT images (Figure 2-9.D); While the smallest calcification size which was visible in the conventional CT was 212-224μm if only the breast phantom A was scanned (Figure 2-9.B). In the clinical, there is no way for a breast to be scanned alone by the conventional CT; chest beneath the breast is always scanned by the conventional CT too. Scanning of chest phantom C (Figure 2-6. C) better

Table 2-1. Visibility qualified as ratio of visible micro-calcifications for various scanning techniques and size range

	180-200μm	200-212μm	212-224μm	212-224μm	224-250μm	250-280μm	KVp	mAs
Conventional CT, chest phantom C				0	0	0	120	400
				0	0	0	80	400
				0	0	0	80	400
Conventional CT, breast phantom A only	0	0	10/25				120	200
	0	0	6/25				120	100
	0	0	0				120	50
	0	0	6/25	4/25	5/25	18/25	80	200
	0	0	4/25	2/25	0	9/25	80	100
	0	0	0	0	0	0	80	50
Cone-Beam CT, breast phantom A	10/25	15/25	14/25				80	27
	8/25	12/25	17/25				80	27
	8/25	15/25	16/25				80	27
	0	10/25	17/25				80	13
	0	14/25	15/25				80	13
	0	15/25	15/25				80	13

represents the real situation. The result showed that none of the calcifications studied (280 μm or smaller) was visible when chest phantom C was scanned with the conventional CT system (Figure 2-6 E,F).

The visibility is quantified as ratio of visible micro-calcifications over whole micro-calcification for various sizes. The results are listed in table 2-1.

2.3.3 Conclusion

Our results indicated that CBCT is capable of imaging calcifications as small as 180 μm. This is due to the intrinsic high spatial resolution of the detector and the application of pendant geometry which allowed the breast to be scanned and imaged alone. The conventional CT is intrinsically lower in spatial resolution. Further disadvantage is the inclusion of the entire chest in the scan and reconstruction which result in degraded spatial resolution as well as overall higher radiation dose to the patient. Our results also showed that in a simulated uniform breast phantom, the embedded location of the micro-calcification is not so important to the visibility.

Chapter 3
Introduction of General Purpose Computation on Graphics Processing Unit (GPGPU) and Compute Unified Device Architecture (CUDA) Environment

GPGPU is the acronym of General Purpose computation on Graphics Processing Units (GPU). GPU is high-performance many-core processors. The latest Graphics Processing Units released by NVIDIA Company — Tesla C1060 including 240 Streaming Processor Cores with frequency of 1.3GHz and 4GB dedicated GDDR3 memory space. The parallel computing power of recent developed GPUs is tremendous; Tesla C1060 is advertised as the world's first teraflop many-core processor, which means the peak value of operations per second of Tesla C1060 is over a teraflop. Such tremendous floating point operation power of GPUs can be used to accelerate a wide range of computationally-intensive or data-immense scientific and engineering computing.

3.1 Introduction of Graphics Processing Units (GPU)

3.1.1 The Predecessor of Graphics Processing Units (GPU)

Graphics Processing Units (GPU) is processors attached to a graphics card. The exploring of special processor for mapping graphics started from 1970s. The first PC video cards were appeared in the early 1980s. In 1990s, 3D real-time graphics were becoming increasingly common on computer games, and professional graphics API such as OpenGL, DirectX was developed, both of which inversely motivated the development

of hardware which make the special graphic processors capable of performing more complex task. In August 1999, Nvidia company coined the acronym GPU (Graphics Processing Unit) when they introduced the NVIDIA GeForce 256 in PC industry, and made the technical definition of a GPU as "a single chip processor with integrated transform, lighting, triangle setup/clipping, and rendering engines that is capable of processing a minimum of 10 million polygons per second." The NVIDIA GeForce 256 was the first PC graphics chip accomplished hardware transform, lighting and shading.

A milestone of the development of GPU was marked in August 2001, Nvidia Company introduced the industry's first programmable GPU, NVIDIA GeForce 3; programmable shading was added to GPUs as new capabilities. Soon after that, similar graphics hardware (programmable GPUs) was introduced by other graphics hardware developers, such as ATI, 3D labs and Matrox. Originally the programmable vertex and fragment shaders were introduced to generate more realistic effect of 3D graphics. Vertex shaders allow the application to program per vertex, which was used to alter vertex attributes, such as color, position, texture coordinates. Programmable fragment shaders (pixel shaders) allows programmer to operate per-fragment during rasterization process. When apply general purpose computation, data were fed to a GPU in the form of textures, and computing programs were loaded as shaders.

3.1.2 Modern GPU

Modern GPUs include hundreds of streaming processors. For example, Tesla C1060 including 240 Streaming Processor Cores. Modern Nvidia chips are based on multiprocessors. Each multiprocessor includes 8 scalar processors and hundreds of ALUs, 8192 (or 16384) registers and 16KB on chip shared memory. Besides, a graphics card

contains a large space global memory with high memory bandwidth, using Tesla C1060 as example, whose memory bandwidth is 102GB/sec. The latest GPUs can outperform orders of magnitude faster than a standard CPU with their high memory bandwidth and floating-point performances. Figure3-1.A[7] shows the growing gap of peak performance between GPU and CPU in recent years, the performance was measured in giga floating point operations per second (GFLOP/s). Figure3-1.B[7] shows the growing gap of memory bandwidth between GPU and CPU.

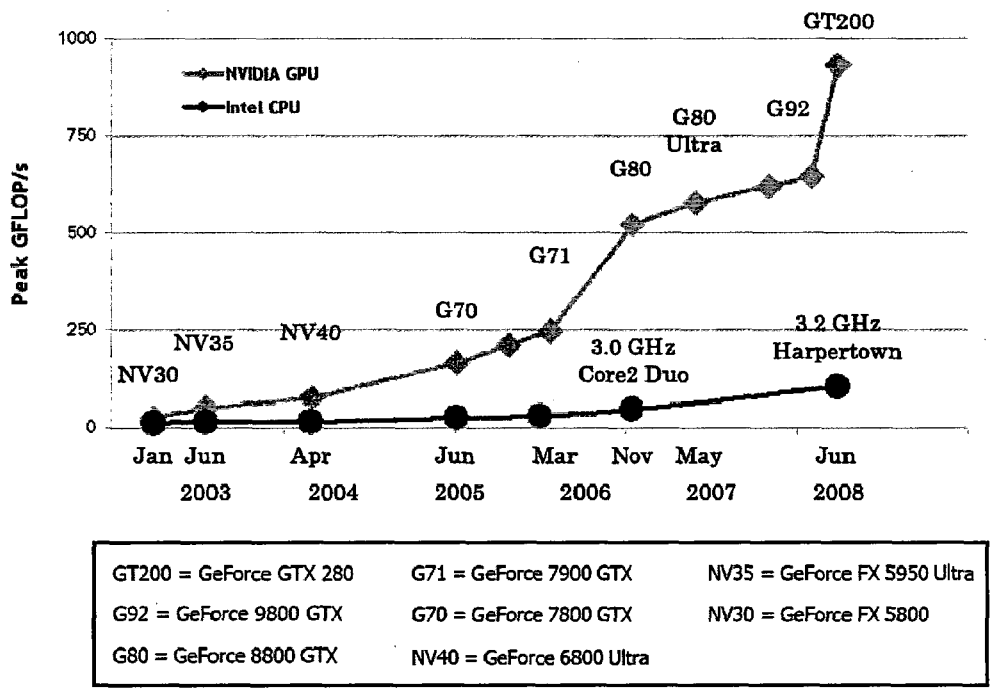


Figure3-1. A. Floating point operation comparison between GPU and CPU.

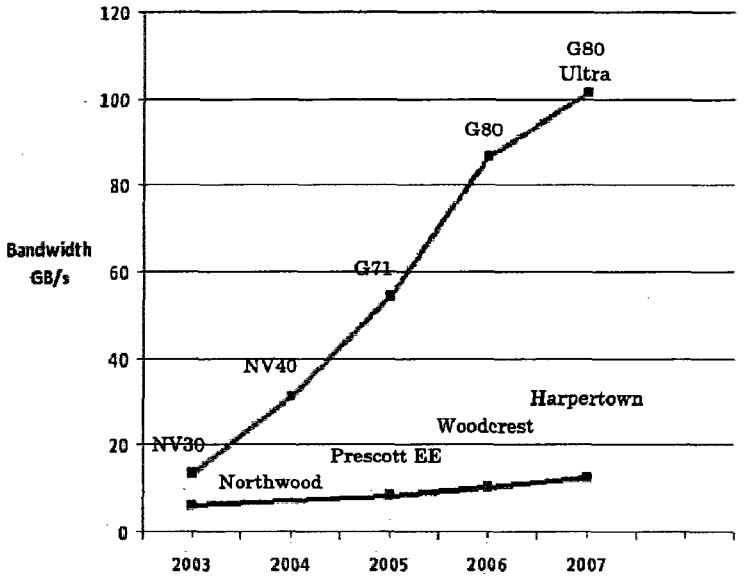


Figure3-1.B Memory bandwidth comparison between GPU and CPU

3.2 Introduction of GPGPU:

For many years, GPU were used to be dedicated graphic rending processors. The introduction of the programmability of the GPU technology intrigued the researchers' interests to exploring the suitability of General-Purpose Computing on Graphics Processing Units (GPGPU). The motivation behind the approach is that GPUs' parallel floating point operations outperform general purpose processors (CPU) by a large margin. The performance gap between the GPU and CPU increases tremendously and the trend is expected to be continuing.

At the beginning, the GPGPU was applied through graphics API such as OpenGL, Direct3D. Compute process is treated as graphics pipeline in these graphics API. In order to achieve general purpose computation through graphics API, programmers have to reorganize their scientific applications to make it looks like graphics application. Data

exchange and storage is inflexibility. Data must be stored in textures which require preliminary packing of large arrays into textures. And writing data to texture is prohibited in GPU. The data output is also inflexibility in graphics APIs, the result data must be written to predefined areas. The other disadvantages of applying general purpose computation through graphics API is that the programming is very complex and the learning threshold is high. The short comes of pursuing GPGPU through graphics API narrowed tremendous application of GPU especially when the hardware of modern GPU had been developed to be more flexible in data access and storage.

People try to develop more specific GPGPU language. Sh, RapidMind, BrookGPU languages are some primary result. Sh is a metaprogramming language. Actually it is a library embedded in C++, dedicated to grogram GPUs together with CPUs for general-purpose computations or graphical applications. Sh was started from 2004 and stop maintenance after August 2006. RapidMind is developed from Sh. Sh is an academic research project, while RapidMind is the commercialized version. The RapidMind's key product is RapidMind Multi-core Development Platform which is a set of C++ libraries providing types and operations used to express parallel computation. Brook and BrookGPU is also a research project developed by Stanford University. As introduced by their website[8], "Brook is an extension of standard ANSI C and is designed to incorporation the ideas of data parallel computing and arithmetic intensity into a familiar, efficient language." They created *streaming* program model. Stream is a data type which can be operated in parallel. The special functions operate stream data in GPU is defined as Kernels.

In November 2006, NVIDIA Company unveiled the CUDA (Compute Unified Device Architecture) language (In Feb 2007, NVIDIA Company released the initial public beta version; In June 2007, they released official 1.0 version), which is an extension of C language with its own compiler and libraries to use GPU for computing. In the CUDA environment, GPU is treated as parallel multiple-cores together with CPU. The development of the CUDA tightly updated with the hardware development of GPU which make it take the full advantage of the flexibility and programmability of new developed GPUs. The introduction of CUDA language boosts the application of GPGPU in various field of scientific computation.

3.3 Difference between CPU and GPU

Central Processing Unit (CPU) or general purpose processor is designed to execute a single thread of sequential instructions with maximum speed. But the increasing of single CPU clock rate is almost hit the wall because of the physics limit, such as the limits of power consumption and the limit of integrated circuit transistor technology, from 2001 to 2003 Pentium 4's clock rate doubled from 1.5 to 3GHz while from 2003 to 2005 the clock rate only increased from 3 to 3.8 GHz. Intel has tried to increase their performance by raise the number of cores. New Intel processors may contain up to four cores. Duo or Quadra CPU cores are designed to use MIMD (multiple instructions / multiple data), various cores execute various instructions, and they work independently.

The development of the GPU is driven by the market demand for real time, high performance 3D graphics which requires huge data parallel arithmetic computing. So GPUs are inherently designed for highly parallel, compute-intensive applications. Most

transistors on GPUs are devoted to data processing; they are organized as arrays of execution units, dispatchers, small volumes of shared memory and memory controllers for several channels. While most CPUs' transistors are devoted to process data caching and flow control. Figure 3-2.[7] shows the different arrangement of GPU and CPU's transistors.

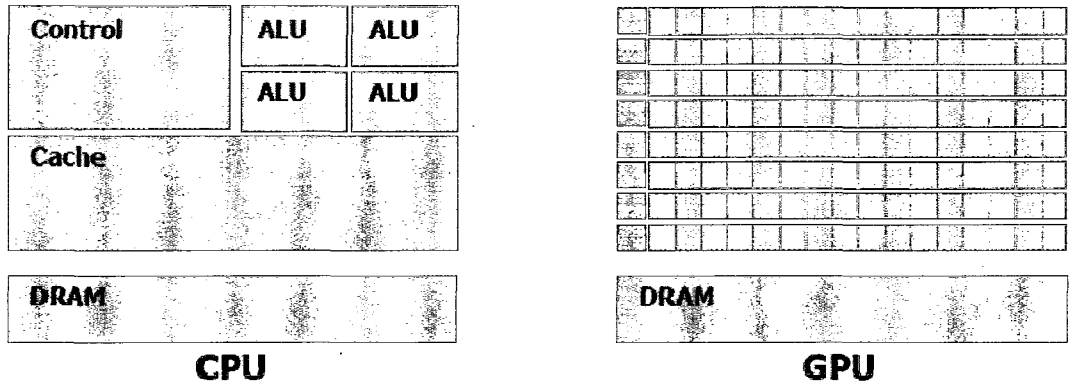


Figure3-2. The difference of transistor devotion between CPU and GPU

In a conclusion, CPU architecture excels at managing many discrete tasks. On the contrary, GPU has great potential for accomplishing huge data, parallel compute-intensive tasks. Until recently, large data-parallel computing domain is mostly occupied by large server clusters and supercomputers which are very expensive to buy and inconvenience to maintain. The development of GPU shows a trend to replace the clusters or supercomputers in some domain.

3.4 Introduction of Compute Unified Device Architecture (CUDA)

CUDA is a general purpose parallel computing architecture. CUDA language combined the advantage of GPUs' parallel computing horse power and CPUs' excellent

execution power of sequential streaming to solve many complex and arithmetic intensive problems in a more efficient way.

CUDA is a software environment based on C language, other languages or application programming interfaces such as FORTRAN, C++, OpenCL, Direct3D and Matlab has already been or will be supported, as illustrated by Figure.3-3[7].

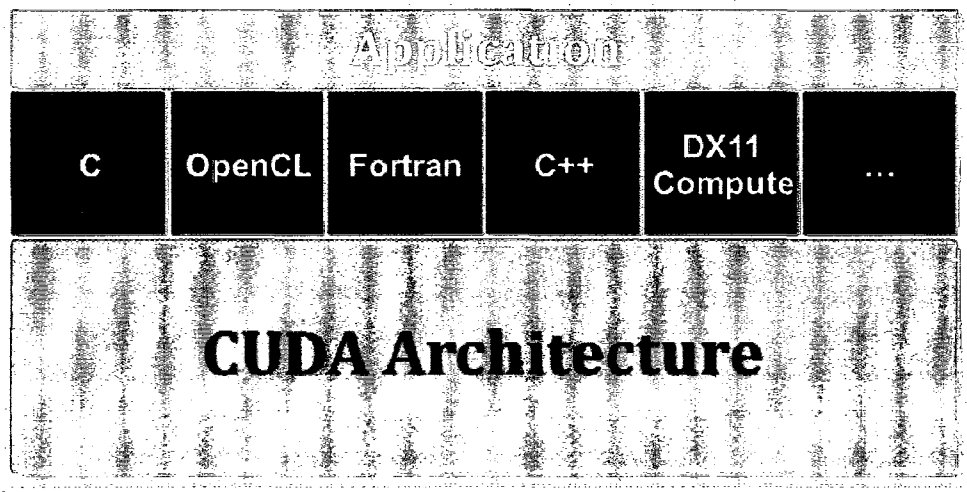


Figure 3-3. CUDA support various languages and application programming interfaces

In the CUDA software environment, all the instructions executed by the CPU are the same as in standard C. The extensions are about the utility of GPUs. The core of the CUDA is three key abstractions related to GPU: a hierarchy of thread groups, memory assignment (shared memory and global memory), and barrier synchronization. In the CUDA environment, problem can be split into sub-problems that can be solved independently in parallel, and the sub-problems can be granulated into small units which can be solved cooperatively in parallel. The architecture of the CUDA provides grained data parallelism and thread parallelism.

3.5 CUDA Programming Model

In CUDA software environment, every function executed by GPUs is defined as a kernel which is declared by a **__global__** specifier. While instructions are executed by a group of parallel threads; the number of threads is claimed inside a <<<...>>> syntax when each time the kernel is called.

3.5.1 Hierarchy of Thread

The hierarchy of the threads is from **thread** (most fine grain) to **block** to **grid**, which can be illustrated as Figure 3-4[7].

Every time a kernel is invocated, a grid of threads is assigned to the kernel. Up to 512 threads can be executed in a block; thousands of same shaped blocks can be executed in the grid in a parallel way. Each of threads is given a unique thread ID within the block which can be accessed by a built-in **threadIdx** variable. Inherited from 3D graphic computation, threadIdx is a 3-component vector. There is a low-latency on chip **shared memory** near each processor core like an L1 cache. Threads within a block can communicate between each other by sharing data through shared memory. And programmer can synchronize the execution in the same block by an intrinsic function **__syncthreads()**. The existence of shared memory and __syncthreads() function make the threads within a block easy to cooperate among themselves. Blocks are organized into a **grid** of thread blocks as illustrated in Figure 3-4. There is a built-in 2-compoment vector variable named **blockIdx** which is used to identify the index of a block within the grid. There are also two built-in variables named **blockDim** and **gridDim** which are used to record the dimension of the block and the dimension of the grid respectively. All the

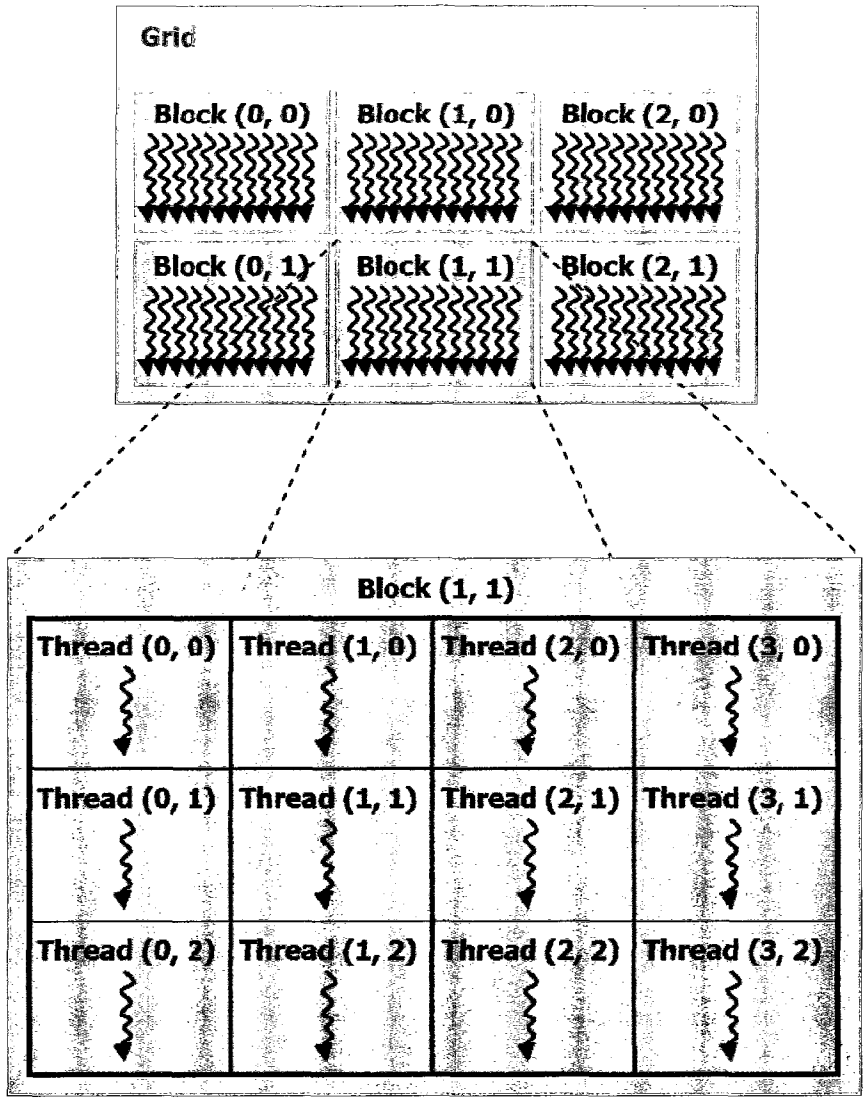


Figure3-4. Threads hierarchy of CUDA

variables ThreadIdx, blockIdx, blockDim and gridDim are accessible within the kernel. The threads in different blocks are required to execute independently. Or in other words, programmer should be careful to code the blocks to make sure that they can be executed in any order.

3.5.2 Hierarchy of Memory:

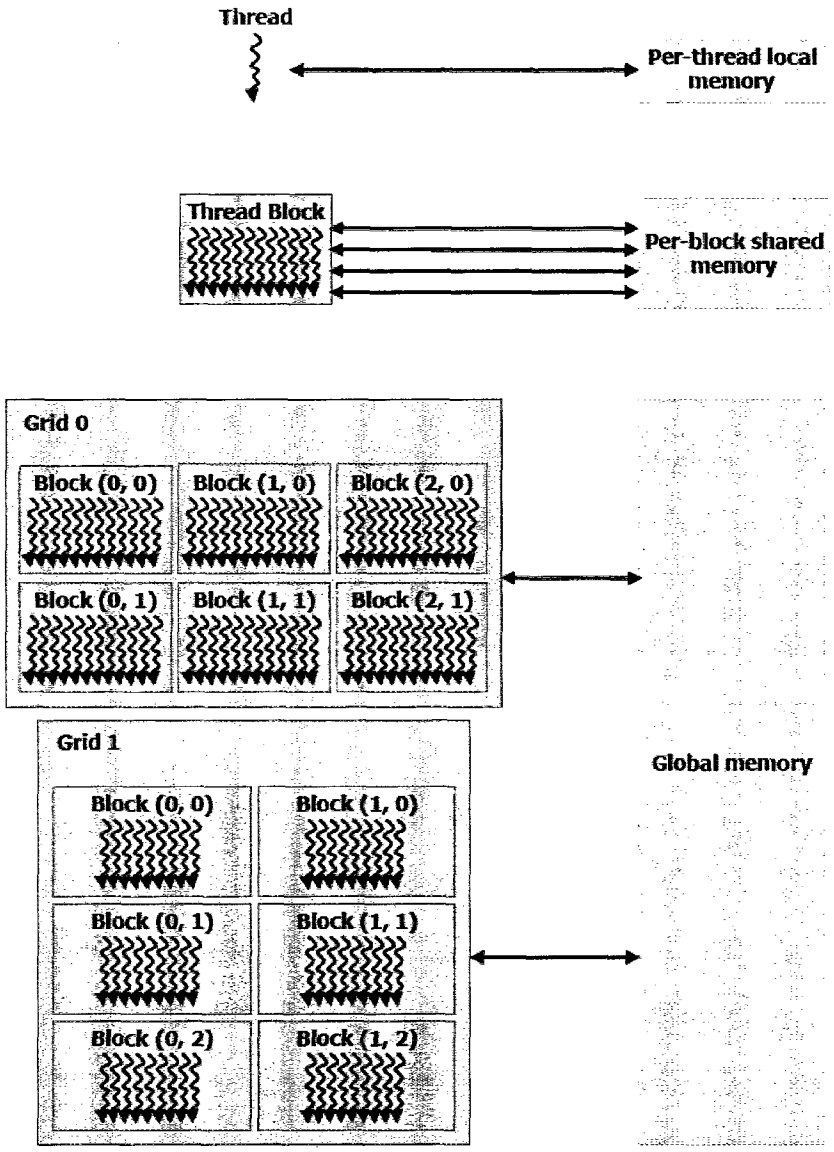


Figure3-5. Memory hierarchy of CUDA.

There are several kinds of memory spaces opened to CUDA threads. First, all threads in a kernel can access to a large space **global memory**. Second, each thread has a private **local memory**. Third, the threads within a same block share a **shared memory**. In fact, shared memory is invocated by a multiprocessor when it loads the blocks (which

will be introduced later). So the shared memory has the same lifetime as the block. Besides, there are also two read-only spaces: the constant and texture memory spaces which are accessible by all threads. Data can be exchanged between GPUs and CPUs by the global, constant, and texture memory. These three kinds of memory can be persistent across kernels. So data in global memory changed by one kernel can be used by the later launched kernels. The memory hierarchy is illustrated in figure3-5[7].

3.5.3 Host and Device

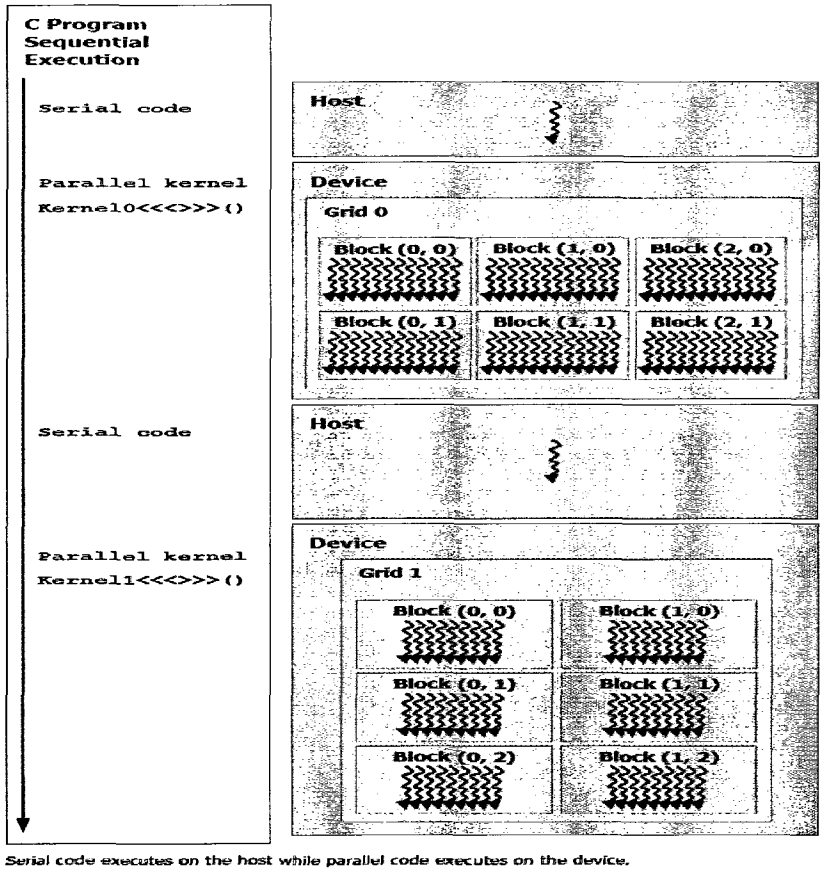


Figure 3-6. The structure of CUDA code

CUDA separates CPU and GPU as **host** and **device**. In CUDA environment, only kernel functions are executed by GPUs, and the rest of the codes (sequential codes) are executed by CPU. The communication and data exchange between CPU and GPU is executed by CUDA runtime. The memory maintained by the host (CPU) is referred to as host memory, while the memory maintained by the device (GPU) is referred to as device memory. Global, local and texture memory are device memories. A normal structure of a CUDA file is illustrated as figure3-6[7].

3.5.4 Hardware Implementation of CUDA

The hardware implementation of CUDA can be illustrated by Figure 3-7[7]. From the view of hardware, the CUDA architecture is based on a set of SIMT (single-instruction, multiple-thread) streaming multiprocessors with on-chip memories. "A multiprocessor consists of eight Scalar Processor (SP) cores, two special function units for transcendental, a multithreaded instruction unit, and on-chip shared memory."[7] when a kernel is invoked, one or several blocks of the grid is distributed to multiprocessors. The threads in a block are grouped by 32 parallel threads which are called **warps**. The multiprocessor SIMT unit selects a warp and issues one common instruction to the warp at a time. If all of the 32 threads of a warp are executing the same path, peak performance can be achieved. Otherwise, if threads within a warp diverge due to conditional branch, the warp serially executes each branch. Different warps execute independently, so branch divergence between different warps will not affect the performance of the GPU. There are four types of on-chip memories maintained by a multiprocessor: a local register per each thread, read-write shard memory shared by all scalar processor cores, read-only constant cache and texture cache that are shared by all scalar processor cores. All the

threads blocks loaded by a multiprocessor shares the register space and shared memory space of the multiprocessor, so the number of blocks a multiprocessor can maintain is limited by the required registers per thread and shared space per block. The kernel will fail to launch if the registers or shared memory required by one block beyond the registers or shared memory that one multiprocessor can provide. CUDA also support multiple GPUs as CUDA devices.

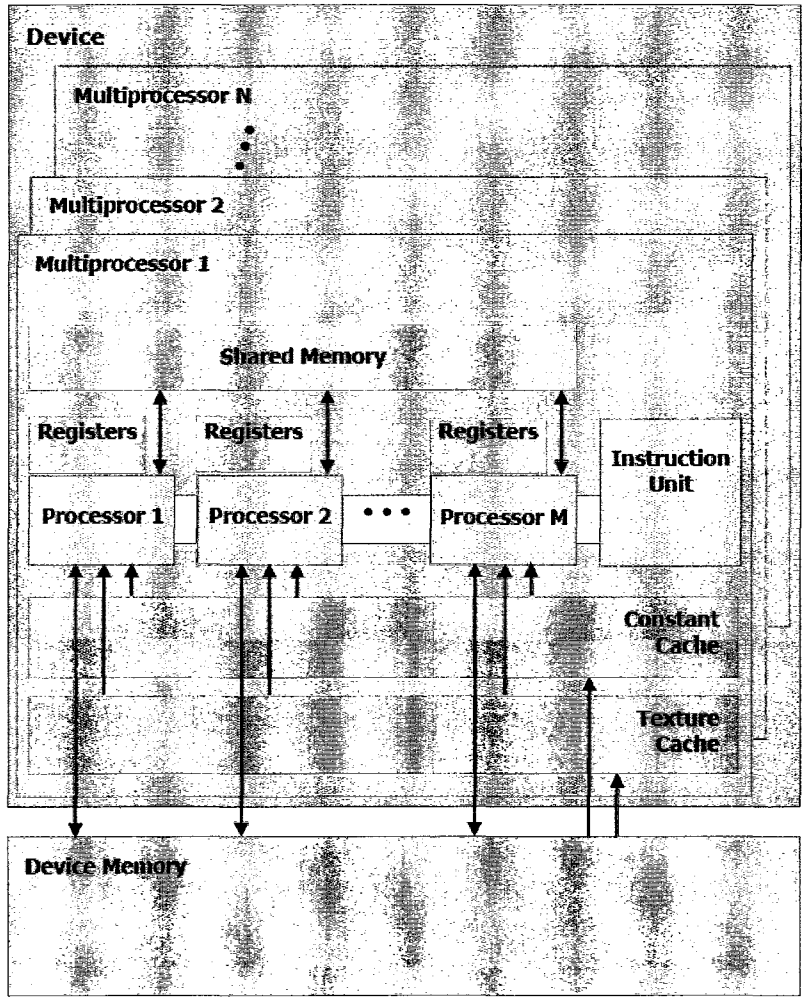


Figure3-7。Hardware implementation of CUDA-a set of multiprocessor

Chapter 4
Improvement of image reconstruction speed with GPU

We will first discuss mathematical basis of tomography with nondiffracting sources. We will show step by step how one can go about recovering the images of the cross section of an object from the projection data. The Fourier Slice Theorem and FDK algorithm applied to reconstruct 3D object in cone beam system are introduced.

In my project, FDK algorithm[6] is used to reconstruct cone beam CT breast imaging. A linux PC cluster system and GPU cards with CUDA environment are used to compute this parallel reconstruction algorithm. For specimen size of 1024x1024x512, over 10 times speed improvement is achieved through the utility of a GPU compared to a PC cluster.

4.1 Three-dimensional image reconstruction

4.1.1 Introduction of the Fourier Slice Theorem

For 2D parallel x-ray projection, considering a *(t,s)* coordinate system to be a rotated version of the original *(x,y)* system as expressed by:

$$\begin{bmatrix} t \\ s \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.1}$$

In the *(t,s)* coordinate system, a projection along line *s* at point *t* is given by:

$$P_\theta(t) = \int_{-\infty}^{+\infty} f(t,s)ds \tag{4.2}$$

And its Fourier Transform is $S_\theta(w)$:

$$S_\theta(w) = \int_{-\infty}^{+\infty} P_\theta(t)e^{-j2\pi wt}\,dt \tag{4.3}$$

The Fourier Slice Theorem relates the Fourier Transform of a projection to the Fourier Transform of the object along a single radial, which can be stated as[9]: "The Fourier transform of a parallel projection of an image $f(x,y)$ taken at angle θ gives a slice of the two-dimensional transform, $F(u,v)$, subtending an angle θ with the u-axis. In other words, the Fourier transform of $P_\theta(t)$ gives the values of $F(u,v)$ along line BB in Figure 4-1."

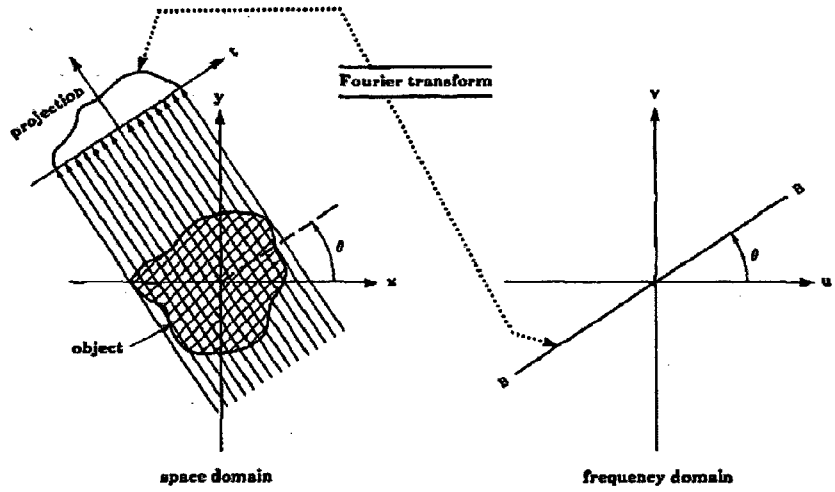


Figure 4-1. Illustration of the Fourier Slice Theorem[10].

According to the Fourier Slice Theorem, $S_\theta(w) = F(w\cos\theta, w\sin\theta) = F_\theta(u,v)$. The 2D object function $f(x,y)$ can be recovered by using the inverse Fourier transform:

$$f(x,y) = \iint_{-\infty}^{+\infty} F(u,v)e^{j2\pi(ux+vy)}\,dudv \tag{4.4}$$

Thus if image projections at enough angles are acquired, it is possible to complete estimate the two-dimensional Fourier Transform of the object *F(u,v)*. And the object can be estimated by applying an inverse Fourier transform on *F(u,v)*. The Fourier Slice Theorem provides a basic conceptual model of tomography.

Filtered back-projection algorithm can be achieved by rewriting the inverse Fourier transform of equation (4.4) in polar coordinates and rearranging the limits of the integration therein:

$$f(x,y) = \int_0^{2\pi}\int_0^{+\infty} F(w,\theta)e^{j2\pi w(x\cos\theta+y\sin\theta)}dwd\theta \tag{4.5}$$

$$f(x,y) = \int_0^{\pi}[\int_{-\infty}^{+\infty} F(w,\theta)|w|e^{j2\pi w(x\cos\theta+y\sin\theta)}dw]d\theta \tag{4.6}$$

$$f(x,y) = \int_0^{\pi}[\int_{-\infty}^{+\infty} S_\theta(w)|w|e^{j2\pi w(x\cos\theta+y\sin\theta)}dw]d\theta \tag{4.7}$$

$$f(x,y) = \int_0^{\pi} Q_\theta\,(x\cos\theta + y\sin\theta)d\theta \tag{4.8}$$

$$Q_\theta(t) = \int_{-\infty}^{+\infty} S_\theta(w)|w|e^{j2\pi w(x\cos\theta+y\sin\theta)}dw \tag{4.9}$$

If the projection data are sampled with a sampling interval of τ cm, the maximum frequency in transform domain is $W = \frac{1}{2\tau}$ cycles/cm. Then equation (4.9) should be expressed as:

$$Q_\theta(t) = \int_{-\infty}^{+\infty} S_\theta(w)H(w)e^{j2\pi w(x\cos\theta+y\sin\theta)}dw \tag{4.10}$$

where $H(w) = |w|b_w(w)$, $\qquad b_w(w) = \begin{cases}1 & |w| < W\\ 0 & otherwise\end{cases}$ (4.11)

The inverse Fourier Transform of *H(w)* is *h(t)*, while the samples $h(n\tau)$ can be given by:

$$h(n\tau) = \begin{cases}\frac{1}{4\tau^2}, & n = 0\\ 0, & n\ even\\ -\frac{1}{n^2\pi^2\tau^2}, & n\ odd\end{cases} \tag{4.12}$$

By the convolution theorem the filtered back projection (4.10) can be written as:

$$Q_\theta(t) = \int_{-\infty}^{+\infty} P_\theta(t')h(t-t')dt' \quad (4.13)$$

The simple equation for parallel beam x-ray image reconstruction can be expressed as:

$$f(x,y) = \frac{1}{2}\int_0^{2\pi}\int_{-t_m}^{t_m} P_\theta(t)h(x\cos\theta + y\sin\theta - t)dt \quad (4.14)$$

4.1.2 Introduction of FDK algorithm

Supposing a ray of the cone beam x-ray flux transport along the direction of a line *u*, a new coordinate system *(h,u,v)* can be obtained after two rotation of the *(x,y,z)*-axes. The first rotation is done in the *x-y* plane, rotating θ degrees around the z-axis to give *(h,u',z)* –axes. Then rotate the *(u',z)* plane around the *h*-axis by an angle of γ to give *(h,u,v)* -axes. In matrix form the rotation are given by:

$$\begin{bmatrix} h \\ u \\ v \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma \\ 0 & -\sin\gamma & \cos\gamma \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (4.15)$$

Feldkamp, Davis and Kress (FDK)[6] describe an approximate reconstruction algorithm for circular cone beam tomography. In the FDK algorithm, cone beam projection data from different angles are filtered and back-projected along x-rays after reconstruction voxel to x-ray source distance and angular differential are properly modified. The value of a voxel is the summed contributions from all horizontally titled fan beams passing through the reconstructed voxel.

In a cone beam system the x-ray source S and flat panel detector are rotated around the rotation axis $O\xi$ in a plane which is perpendicular to $O\xi$, as shown in Figure 4-2. The axis *OS* always passes through the x-ray source and is perpendicular to the detector plane and intersects at D. The rotation angle of the axis *OS* with respect to its original position

is denoted as β. O_β represents the rotating coordinate system *(t,s,z)*. Compare to original coordinate *(x,y,z)*, *t,s* can be expressed as:

$$t = x\cos\beta + y\sin\beta \tag{4.16}$$

$$s = -x\sin\beta + y\cos\beta \tag{4.17}$$

Each x-ray beam in the rotating coordinate system can be identified by four variables p', ξ', γ, κ. Here p' and ξ' denote the horizontal and vertical coordinates of the point in the plane of detector, γ represents the angular location of a ray, called fan angle, and κ represents the location of a tilted fan, called cone angle.

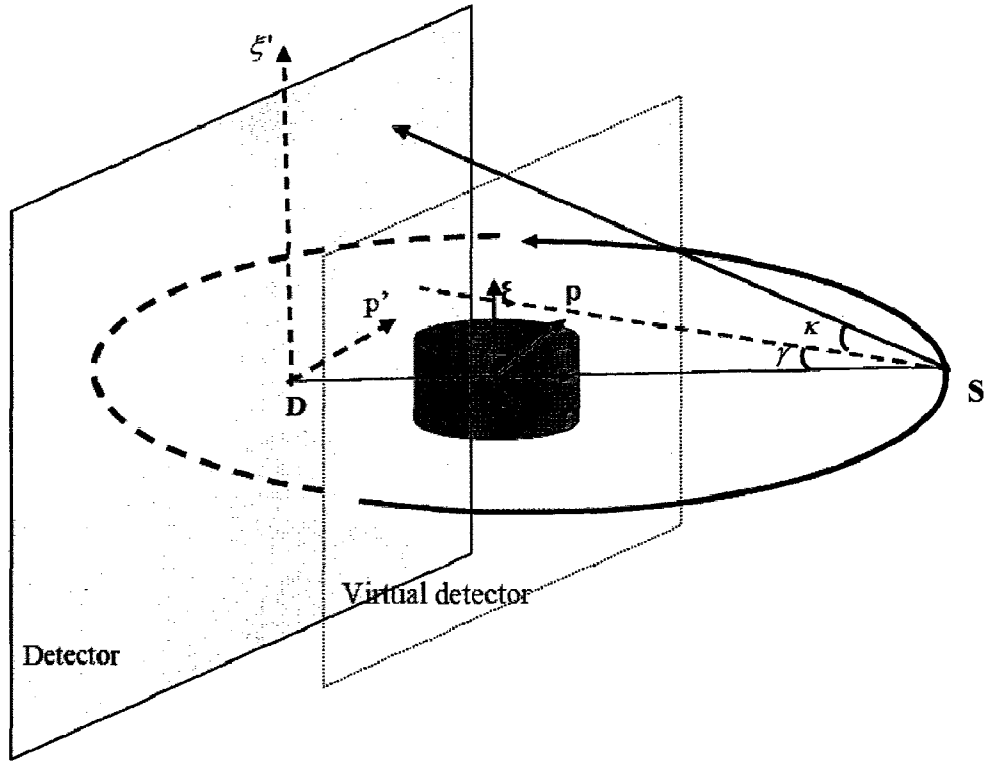


Figure 4-2. Cone beam scanning geometry of rotation coordinate system O_β. X-ray source and detector circular with radius of d_{SO} and d_{SD} respectively around the rotation axis $O\xi$.

The actual detector is not essential for cone beam projection and reconstruction[10], therefore we assume a virtual detector plane *(p, ξ)*, parallel to the actual detector plane

(p', ξ'), containing the rotation axis $O\xi$. According to the cone beam geometry magnification, virtual detector coordinates is given by multiplying actual detector coordinates with D_{SO}/D_{SD}. Here D_{SO} and D_{SD} denote the distance from the x-ray source to the rotation center and the distance from the x-ray source to the detector, respectively.

$$p = p' \frac{D_{SO}}{D_{SD}}, \qquad \xi = \xi' \frac{D_{SO}}{D_{SD}} \tag{4.18}$$

The fan angle γ and cone angle κ can be defined as:

$$\gamma = \tan^{-1}\left(\frac{p}{d_{SO}}\right) \tag{4.19}$$

$$\kappa = \tan^{-1}\left(\frac{\xi}{\sqrt{d_{SO}^2+p^2}}\right) \tag{4.20}$$

The cone beam reconstruction algorithm can be broken into three steps:

Step 1:

Prior to convolution with a ramp filter, a pre-weighting factor, depending on both the fan angle and cone angle, is applied on the projection data, yielding:

$$P'_\beta(p,\xi) = P_\beta(p,\xi)\frac{d_{SO}}{\sqrt{d_{SO}^2+p^2+\xi^2}} \tag{4.21}$$

$$\frac{d_{SO}}{\sqrt{d_{SO}^2+p^2+\xi^2}} = \frac{d_{SO}}{\sqrt{d_{SO}^2+p^2}}\frac{\sqrt{d_{SO}^2+p^2}}{\sqrt{d_{SO}^2+p^2+\xi^2}} = \cos\gamma\cos\kappa \tag{4.22}$$

Where $P_\beta(p,\xi)$ represents the projection data at rotation angle β. The pre-weighting factor is geometrically interpreted as the cosine of the angle between the ray and central ray of the projection. From the equation, we can see the weighting factor independent of rotating angle β.

Step 2:

Convolve the weighted projection $P'_\beta(p,\xi)$ with $h(p)/2$ by multiplying their Fourier Transforms with respect to p. This convolution is done independently for each elevation ξ:

$$Q_\beta(p,\xi) = P'_\beta(p,\xi) * \frac{1}{2}h(p) \tag{4.23}$$

And here $\xi = \frac{D_{SO}z}{D_{SO}-s}$.

Step3:

Finally, each weighted projection is back-projected over the 3D reconstruction grid:

$$g(t,s,z) = \int_0^{2\pi} \frac{D_{SO}^2}{(D_{SO}-s)^2} Q_\beta\left(\frac{D_{SO}t}{D_{SO}-s}, \frac{D_{SO}z}{D_{SO}-s}\right) d\beta \tag{4.24}$$

$$f(x,y,z) = g(t(x,y), s(x,y), z) \tag{4.25}$$

Here $t(x,y)$, $s(x,y)$ are equations of (4.16) and (4.17) respectively.

4.2 Parallel Computing Implementation

4.2.1 Device Specification

A Linux PC cluster is used in our cone beam CT reconstruction and simulation study. Several Nvidia Graphic cards are also used in our study to compare the speed improvement of utilizing GPU. Two Nvidia Tesla C1060 and one Nvidia Geforce 8800 GTS graphic cards have been utilized in our study.

Here is the device specification of Linux PC cluster system. There are 64 computing units (Opteron 2214 HE computing processors) in the PC Cluster. Each unit has 2 giga-bytes (GB) of memory. Theyall run the Linux operating system with kernel 2.4.18, and the standard open source supporting software packages such as GNU compliers are installed in each computing unit. The most frequently used communication libraries for parallel computation such as MPI, OpenMP, PVM, and Pthreads, and

resource management and scheduling software such as OpenPBS, Maui, and Condor are all compiled and integrated into the system. The FDK algorithm has been coded in C language[11] and MPI library was adopted for image reconstruction. Since the filter operations on one projection image are independent of other projection images, they were decided to parallelize by distributing equal number of projections to each processor.

A Nvidia Geforce 8800 GTS graphic card combined with a supporting CPU and a (or two) dedicated general purpose computing card Nvidia Tesla C1060 combine with a supporting CPU (Intel (R) Core™ i7,CPU 940 @2.93 GHz) are used to achieve high speed parallel computing of imaging reconstruction. The compute capability of Geforce 8800 GTS is 1.0. Geforce 8800 GTS has 12 multiprocessors and each multiprocessor includes 8 scalar processors and hundreds of ALUs, 8192 registers and 16KB on chip shared memory. The clock rate is 500 MHz (or 1.3GHz). Geforce 8800 GTS has 320 MB global memory with memory bandwidth 64 GB/sec. But Geforce 8800 GTS also serve as display graphic card, so only approximately 200 MB's global memory can be reserved as general purpose computing usage.

Tesla C1060 is one of the super-fast computing graphic cards; it is not a video card any more, it is dedicated on computation. The compute capability of Tesla C1060 is 1.3. Tesla C1060 has 30 multiprocessors (240 computing cores) with 16384 registers for each multiprocessor. The processor's clock rate is 1.3GHz. One of the other advantages of Tesla C1060 is its huge global memory, 4GB DDR3 memory size with102 GB/sec memory bandwidth and 800MHz memory speed. And there is no run time limit on Tesla C1060.

Paying attention to some specifications and features (as show in table 4-1) of GPU helps us improve the performance of our parallel computing. As illustrated in Chapter 3, there are many kinds of memories in GPU; carefully utilizing different kind of GPU memory is important. In a summary, there are global memory, shared memory, local memory, and constant memory and register memory. Data in global memory, shared memory are declared and saved by user. Constant variable, parameters of kernel function are saved on constant memory. Automatic variables declared in device code (kernel) resident in registers. Throughput of memory operations is 8 operations per clock cycle. When accessing local memory or global memory, there are, in addition, 400 to 600 clock cycles of memory latency. Much of this global memory latency can be hidden by the thread schedule if there are sufficient independent arithmetic instructions that can be issued while waiting for the global memory access to complete. Shared memory is user-managed L1 cache, and is much faster to access than global memory (4 clock cycles vs. 400-600 memory latency). But in our code, the memory size of shared memory is too small for our data, so we didn't use shared memory. Generally, accessing a register is zero extra clock cycles per instruction. It is better to put multiple time called variables in registers to save time. But delay may occur due to register memory bank conflicts. "The compiler and hardware thread scheduler will schedule instructions as optimally as possible to avoid register memory bank conflicts. They achieve the best results when the number of threads per block is a multiple of 64. Other than following this rule, an application has no direct control over these bank conflicts."[12]

Some other specifications and features of Nvidia GPU with compute capability of 1.0 (Geforce 8800 GTS) and 1.3 (Tesla C1060) are show in table 4-1.

Table 4-1. Specifications of Nvidia GPU with compute capability 1.0 and 1.3[7].

Compute capability	1.0	1.3
The maximum number of threads per block	512	512
The warp size	32	32
The number of registers per multiprocessor	8192	16384
The amount of shared memory available per multiprocessor	16KB	16KB
The maximum number of active blocks per multiprocessor	8	8
The maximum number of active warps per multiprocessor	24	32
The maximum number of active threads per multiprocessor	768	1024

4.2.2 Projection Data Acquirement

A post-mastectomy breast put in a soda bottle was scanned with a bench-top experimental cone-beam breast CT (CBCT) system, as show in Figure 4.3. The system consists of a conventional tungsten target x-ray tube (G-1592, Varian Medical Systems) and a 30x40 cm^2 amorphous silicon/cesium iodide (a-Si:H/CsI) flat panel (FP) detector (Paxscan 4030CB, Varian Medical Systems). The focal spot size is 0.6mm and the pixel size of the flat panel is 194 μm. The pixel number in y direction and z direction of the detector are 2048 and 1536 respectively. A motor-driven rotation stage is used to position and rotate the specimen to simulate dedicated breast CT in which the patient would lie on a table in the prone position with one breast drawn downward thought an opening to allow the x-ray tube and detector to rotate around and scan the breast beneath the table.

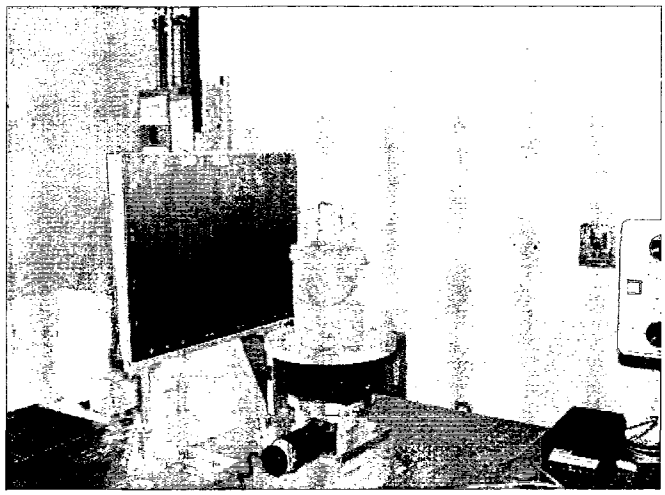


Figure 4-3. A stationary gantry, rotating phantom bench top system was used to acquire projection images of a post-mastectomy specimen. The post-mastectomy breast held by a cola bottle was put on the rotation stage.

The source to detector and source to iso-center distances are 104 and 75 cm respectively. When scan is performed, 300 projection images over 360° were acquired at 80kvp, 25mA with 7.5 frames per second. Whole projection data is acquired within 40 seconds.

4.2.3 Parallel computing implementation

In cone beam breast CT system, projection data should be calculated as

$$P_\beta(p,\xi) = -\lg\left(\frac{I_\beta(p,\xi)}{I_o}\right) \tag{4.26}$$

Here $I(p,\xi)$ is photon intensity after x-ray pass through object, and I_o is the original x-ray intensity. And object specimen $f(x_i,y_j,z_k)$ represents attenuation value of tissue in each voxel. In both GPU and PC cluster system, FDK algorithm is applied to reconstruction image.

4.2.4 Parallel computing with PC cluster

In PC cluster, the computing was parallelized by distributing equal number of projections to each processor. If there are 30 nodes applied in one compute processing for the 300 projection images, each node deals with 10 projection images. Each processor calculates the weighted projection data, convolves the each line of weighted projection data with *h(p)/2* and sums the contribution from these 10 projection images to every voxel of the specimen. At the final stage, all the contributions for the volume voxels of the specimen from each processor are loaded to and summed by the master node.

4.2.5 Parallel computing with GPU

In my project, one Nvidia Geforce 8800 GTS graphic card, one Nvidia Tesla C1060 or two Nvidia Tesla C1060 were separately utilized as device to calculate image reconstruction. When GPU is applied as parallel computing device, four kernels were built for the image reconstruction. CUDA provide CUFFT library to perform the Fast Fourier Transform and inverse Fast Fourier Transform.

The weighting factors are independent of rotation angle β, so weighting factors are same for projection image at each angle. I built the first kernel named *weightKernel* (code is attached in Appendix A) to calculate the array of weighting factors. Each weighting factor is calculated by a thread. Respecting to the 2048x1536 pixel size of detector, the size of the weighting factors' array should also be 2048x1536. The big array with size 2048x1536 is equally separated in to a lot of small array with size 128x4, in the CUDA environment. There are two 3-component build-in variables named blockDim and gridDim which are used to record the dimension size of block and grid respectively. When launching the *weightKernel*, the block size is assigned to be 128x4. It is the result of considering two reasons: one is that the maximum number of threads per block is 512,

the other is that blockDim.x 128 is the multiple of warp size 32, which will help to avoid warp diverges and speed up data access and calculation.

Second kernel is named *calmiu_kernel* (Appendix A) which is used to calculate weighted projection data $P'_\beta(p,\xi)$. First projection data is calculated by equation (4.26). Then weighted projection data $P'_\beta(p,\xi)$ is calculated by multiplying projection data $P_\beta(p,\xi)$ with weighting factor at each point. When launching the kernel, the data of same pixel at different angle are located in the same block, leading to block size of 300. The grid size is the same as detector size 2048x1536.

The next step is to convolve the weighted projection data with ramp filter. The Fourier Transform of Ramp filter *h(p)/2* is calculated through the function provided by CUDA library CUFFT; we can express this operation as FFT(*h(p)/2*). And each line of the projection data is calculated through CUFFT too; we can express this operation as FFT($P'_\beta(p,\xi)$). In order to get fast and accurate result, complex Fast Fourier Transform is performed. And the data is saved as complex type in global memory. A third kernel named *convolution_kernal* (Appendix A) is launched to multiply the result of FFT(*h(p)/2*) with that of FFT($P'_\beta(p,\xi)$) at each point. As in *calmiu_kernel*, the block size is 300 for the data of each angle, and the grid size 2048x1536, which leading to only one filter data is needed to load into a block.

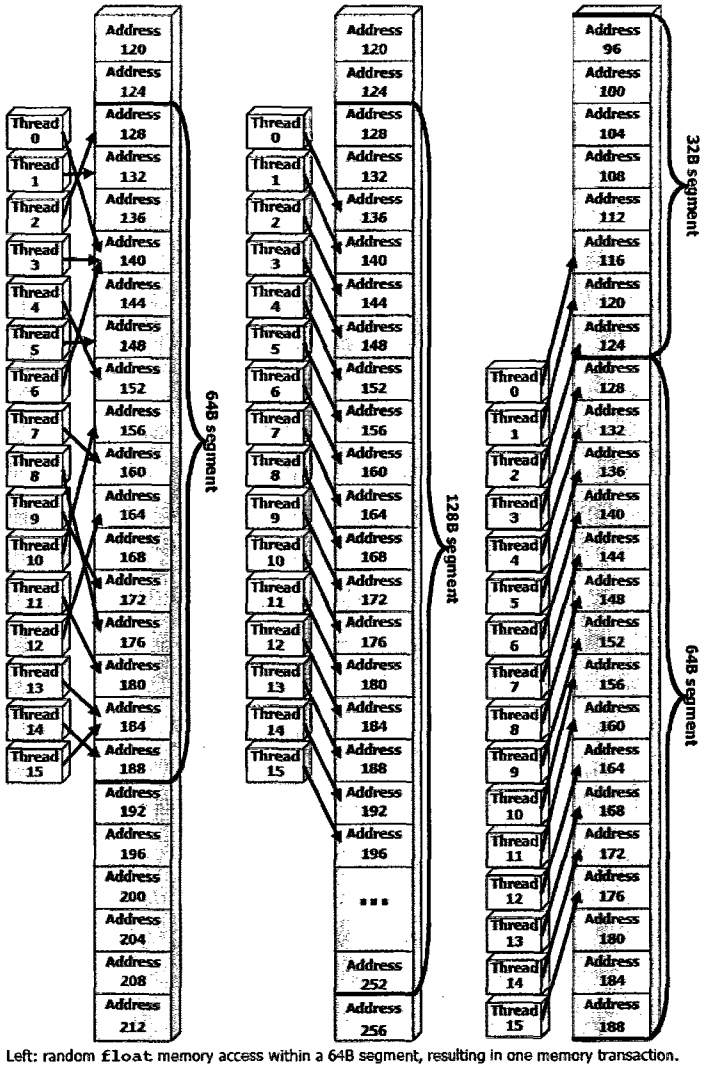
The fourth kernel named *back_kernal* is used to back-project filtered projection data to reconstruction image. This is the most time consuming kernel. Most of the time consumption of image reconstruction is due to back-projection process. In order or get better performance, we need to carefully schedule threads assignment within GPU. In the kernel, 14 registers are used per thread. In my Tesla C1060 code, 128 threads per block

are assigned in the kernel. This also lead to GPU occupancy data as: active threads per multiprocessor 1024, active warps per multiprocessor 32, active blocks per multiprocessor 8. Active registers per multiprocessor is 14336, less than 16384 which is the number of register a multiprocessor can support. In my Geforce 8800 code, 64 threads per block are assigned in the kernel. This also lead to GPU occupancy data as: active threads per multiprocessor 512, active warps per multiprocessor 16, active blocks per multiprocessor 8. It should be the same to choose any value which is multiple of 64 and less than 512 as number of threads per block. I have ever tried to schedule 64, 128, 256, or 512 threads to each block when launch back-projection kernel (Tesla C1060); results showed that the performance of the kernel is same. As illustrated in Chapter 3, threads launched in multiprocessor are grouped as warps. Threads in the same warp simultaneously execute the same instructions. Thread number in a block is the multiple of warp size can fully use the parallel computing power of scalar cores in the multiprocessor. It is helpful to hide memory access latency, when maximum warps or half maximum warps per multiprocessor are launched in the kernel. Choosing number of threads within a block as multiple of 64 also help to avoid register bank conflicts. The voxel value of specimen is calculated by adding contribute from 300 views which is achieved by a loop in the kernel. Avoiding register bank conflicts may be important in this situation, since registers inside the loop will be executed 300 times.

Global memory bandwidth is used most efficiently when the simultaneous memory accesses by threads in a half-warp can be coalesced into a single memory transaction of 32, 64, or 128 bytes, as show in Figure 4-4. For the device with compute capability of 1.2 and higher, coalesced memory access in global memory can be simply illustrated as:

Address in the global memory can be segmented by 32, 64, or 128 bytes. If threads within the same half warp access the same segment, only one transaction is required. Otherwise, the number of segment the threads need to access decides the number of transaction required. Inappropriate global memory access can lead to huge performance decrease.

In the Tesla C1060 back projection kernel, different global memory access are tried. The projection data are saved row by row (first p'-direction, then ξ'-direction); and the result specimen CT value image is also saved row by row (first y-direction, then x-direction, then z-direction) in global memory. In the first method, threads in a block calculate voxels within the same row (y-direction). In this situation, 16 threads in the same half-warp read projection data within several 64 bytes segments (single precision floating point data is used in the code, 64 is the product of 4 bytes times 16), which leads to one or several memory transaction. Threads in the same block save CT data in continues address, and16 threads in the same half-warp write ct data within one 32 bytes segment (CT data saved in the type of short int, 32 is the product of 2 bytes times 16), which leads to only one memory transaction. While in the second method, threads in a block calculate voxels along the same column (z-directions). 16 threads in the same half-warp read or write data in 16 address segments which are separated far away from each other. The second method heavily lagged down computer speed.



Left: random `float` memory access within a 64B segment, resulting in one memory transaction.
Center: misaligned `float` memory access, resulting in one transaction.
Right: misaligned `float` memory access, resulting in two transactions.

Figure 4-4. Illustration of global memory access by device with compute capability of 1.2 and higher. Address in the global memory can me segmented by 32, 64, or 128 bytes. If threads within the same half warp access the same segment, only one transaction is required for the whole half warp threads. The number of segment the threads need to access decides the number of transaction required.

I have also tried multiple-GPU (two Tesla C1060) as devices to reconstruct image to see the benefits.

4.3 Result and conclusion

Different compute devices are used to reconstruct images. 300 projection images over 360° were acquired at 80kvp, 25mA when scanning post mastectomy breast specimen. And 1024x1024x512 volume pixels of breast specimen were reconstructed. The time consumption of utilize different device to reconstruct image is show in table 4-2.

Table 4-2. Time consumption of different device

Device	PC Cluster	Geforce 8800 GTS	Tesla C1060
Time (second)	1534.3 (1944.3)	793.3	160.6

It cost PC luster (30 nodes or 30 CPU) 1534.3 seconds user time (the real time it took is 1944.3 seconds) to reconstruct image. It cost Geforce 8800 GTS 793.3 second to reconstruct image. And it cost Tesla C1060 160.6 second to reconstruct image. So Geforce 8800 GTS and Tesla C1060 had 1.9 times or 9.6 times performance gain over PC cluster.

3D image reconstructed by Tesla C1060 and by PC cluster are showed in Figure 4-5 and Figure 4-6 respectively. Image display software is MRIcro, parameters of brightness and contrast are set as -200 and 1000. Both of them are high quality image, Micro-calcifications can be clearly seen in the image. Figure 4-7 shows CT value difference between 3D images reconstructed by GPU and by PC Cluster. The strip artifacts come from the PC cluster results, Figure 4-8 A. and B. shows the same slice of the breast image reconstructed by GPU and PC cluster respectively (brightness :-1000; contrast: 1000).

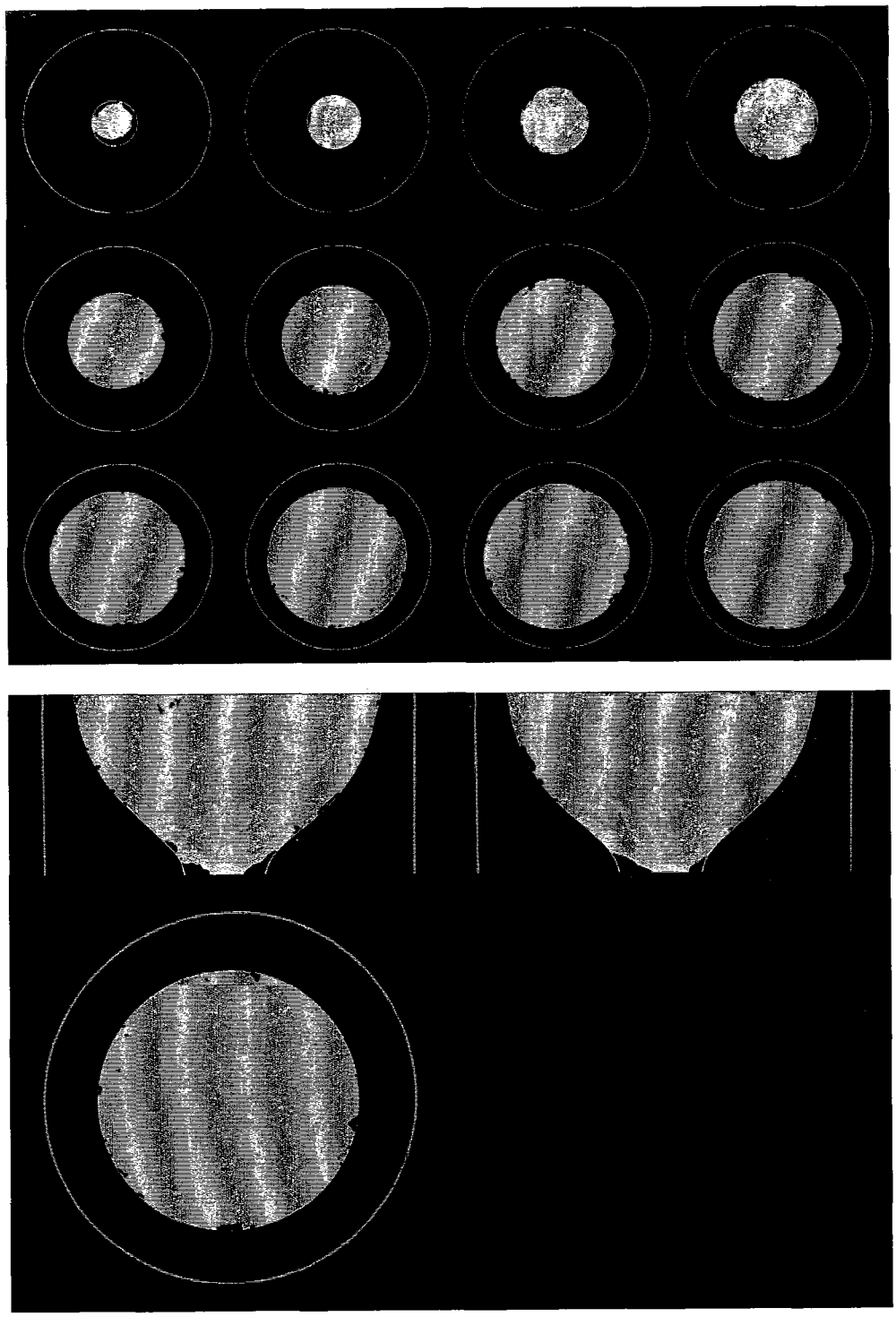


Figure 4-5. Transverse, sagittal and colonal views of breast image reconstructed by Tesla C1060

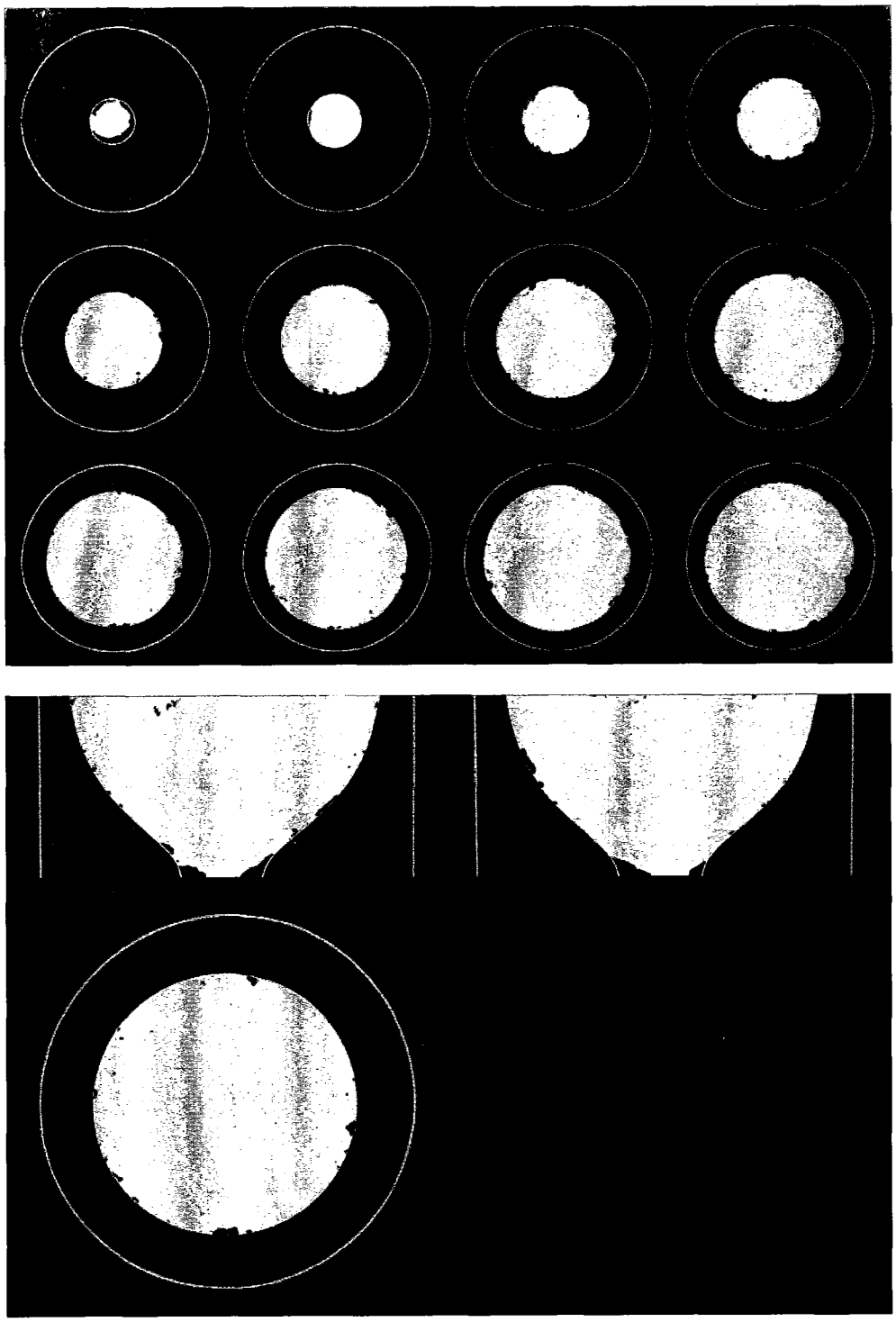


Figure 4-6. Transverse, sagittal and colonal views of breast image reconstructed by PC cluster.

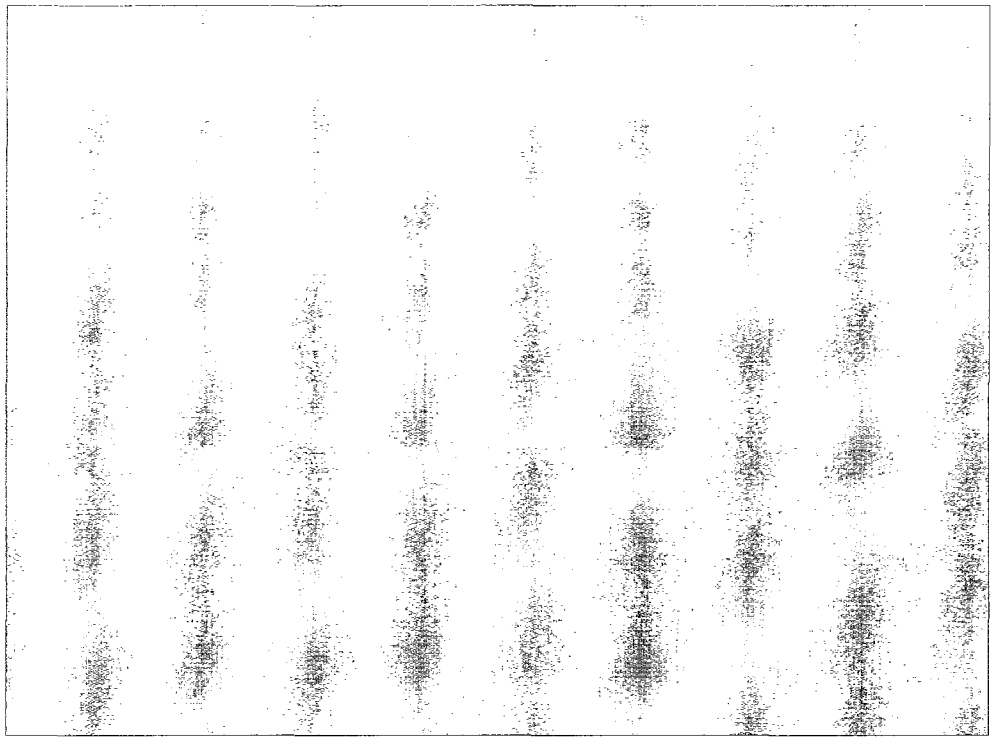


Figure 4-7. CT value differences between 3D images reconstructed by GPU and by PC Cluster.

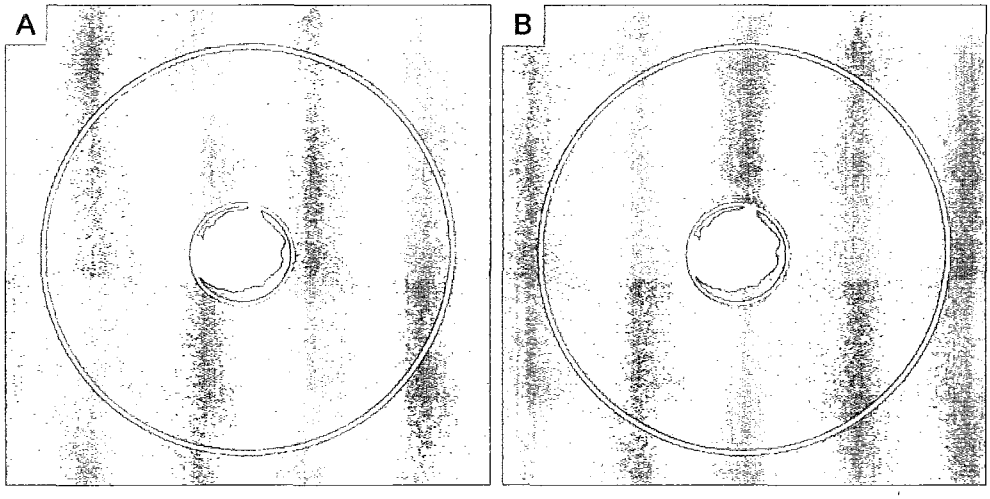


Figure 4-8. A. Transverse view of breast image reconstructed by GPU; B. Transverse view of breast image reconstructed by PC cluster. There are some strip artifacts in the image reconstructed by PC cluster.

Clearly, strip artifacts come from the images reconstructed by PC Cluster. If we let I_{ijk}, I'_{ijk} represent CT value of breast image reconstructed by GPU and by PC cluster. Applying the equation (4.27) to the breast image *(M, N and P are image dimension size 1024,1024 and 512)*, we get result 0.1667. This data is a little large, may be due to the strip artifacts of PC Cluster.

$$\sqrt{\frac{1}{MNP}\sum_{i,j,k}\frac{(I_{ijk}-I'_{ijk})^2}{\left[\frac{1}{2}(I_{ijk}+I'_{ijk})\right]^2}} \tag{4.27}$$

I also investigated the effect of global memory access to the performance of GPU. When launching back-projection kernel on Tesla C1060. Threads in the same block calculate voxel values of the same row or of the same column. As illustrated in section 4.2.3, a block calculates voxel values by row leads to much less global memory transaction than by column. The time consumption for the back-projection process (kernel) are 79.516 second (row by row) and 1986.579 second (column by column) respectively, which shows that inappropriate memory access will lead to heavy lag down of GPU performance.

Multiple-GPU application is also explored in my project. Since back-projection is the most time consumption process, back-projection kernel has been launched to one Tesla C1060 or two Tesla C1060. Time consumption listed in table 4-3.

Table 4-3. Time consumption of one or two GPU

Device	Back-projection (s)	Whole image reconstruction (s)
One Tesla C1060	80.6	135.703
two Tesla C1060	38.9	79.733

From the result of table 4-3 we can see that applying two GPU saves half time for the back-projection process. But data read in and write out to hard disk can only be done by CPU, so utility of two GPU can not save half time for the whole image reconstruction.

All the results above show us that GPU can dramatically improve the image reconstruction speed compared to PC cluster.

There are other advantages of applying GPU than PC cluster to perform image reconstruction. First, GPU is much cheaper than PC cluster. Now, it is $1299.99 for two Tesla C1060; and it is less than $250 for a Geforce 8800 GTS. Second, GPU is much easy to maintain. It is hand size graphic card. Geforce series GPU can be easily plug in Normal desk top computer. Although, Tesla C1060 need special motherboard due to power supply requirement. It is possible to directly install GPU in CBCT system in future to make the system more compact. While, for PC cluser, it is required to set a special room with constant temperature and humidity level, stable power supply to maintain the PC cluster which is inconvenience and cost lots. Third, in reality, PC cluster is separated in a special room, so Much more data transfer time are required for PC cluster system.

Reference

1. *"American Cancer Society: breast cancer"*, 2009.
2. J. M. Boone, T. R. Nelson, and K. K. Lindfors, *"Dedicated breast CT: Radiation dose and image quality evaluation"*, Radiology **221**, 657–667, 2001.
3. B. Chen and R. Ning, *"Cone-beam volume CT breast imaging: Feasibility study"*, Med. Phys. **29**, 755–770, 2002.
4. S. Glick, S. Vedantham, and A. Karellas, *"Investigation of optimal kVp Settings for CT mammography using a flat panel detector"*, Proc. SPIE **4682**, 392–402, 2002.
5. L. Chen, C. C. Shaw, et.al, *"Cone-beam CT breast imaging with a flat panel detector-a simulation study"*, Proc. SPIE **5745**, 943–951, 2005.
6. L.A. Feldkamp, L.C. Davis, and J.W. Kress, *"Practical cone-beam algorithm"*. J. Opt. Soc. Am., 1984. 1: p. 612-619.
7. Nvidia Company, *"NVIDIA CUDA™ Programming Guide—version 2.3"*, 2009.
8. http://graphics.stanford.edu/projects/brookgpu/lang.html
9. KAK. AC, *''Tomographic imaging with diffracting and non-diffracting sources,"* in Array Signal Progessing, S.Haykin, Ed. Englewood cliffs, NJ: Prentice-Hall, 1985.
10. KAK. A.C. and M. Slaney, *"Principle of computerized tomographic imaging"*. New York: IEEE Press, 1988.
11. Post doctor fellow Lingyun Chen wrote the code of 3D imaging reconstruction on PC cluster.
12. Nvidia Company, *"NVIDIA CUDA C Programming Best Practices Guide—CUDA Toolkit 2.3"*, 2009.

Appendix A

```
__global__ void weightKernel( float* d_weight)
{
    float z_prime, y_prime;
    const unsigned int width = blockDim.x*gridDim.x;
    unsigned int idy = blockIdx.x * blockDim.x + threadIdx.x;
    unsigned int idz = blockIdx.y * blockDim.y + threadIdx.y;

    z_prime=-((float)idz- (float)Z_CENTER)*sample_interval_z;
    y_prime= ((float)idy- (float)Y_CENTER)*sample_interval_y;
    d_weight[idz*width+idy] = SO/sqrt(SO*SO + y_prime *y_prime
                                          + z_prime*z_prime);

}


__global__ void  calmiu_kernel(float *d_miu, float *d_weight,
                               float *d_open)
{
    int    id = blockIdx.x*gridDim.y*blockDim.x
              + threadIdx.x*gridDim.y + blockIdx.y;
    float  weight = d_weight[blockIdx.x*gridDim.y + blockIdx.y];

    d_miu[id] =  -log(d_miu[id]/d_open[threadIdx.x])*weight;
}


__global__ void convolution_kernal( Complex* d_miu, Complex *d_filter)
{
    long long int id ;
    float    filter = d_filter[blockIdx.y].x;

        id = blockIdx.x*blockDim.x*gridDim.y + threadIdx.x*gridDim.y
             + blockIdx.y ;
        d_miu[id].x = d_miu[id].x*filter/(1.0*ZP);
        d_miu[id].y = d_miu[id].y*filter/(1.0*ZP);
}


__global__ void back_kernal( float* d_miu, short int* d_ct,int flag,
                             int ib, int z_slice)
{
    float        temp,z,t,s,x,y,z_o, so_s;
    int          id,I,k,z_int,t_int;
    float        belta_s = -belta_step*PI/180.0;

     x   = (blockIdx.y - REC_XY_CENTER)*recon_step;
     z_o = ((int)blockIdx.x + ib*z_slice + Z_CENTER - zstart)
           *recon_step_z;
```

```
        for(k=0;k<flag;k++){
            id = blockIdx.x*blockDim.x*flag*gridDim.y
                 + blockIdx.y*blockDim.x*flag + k*blockDim.x + threadIdx.x;
            y  = (k*blockDim.x + (int)threadIdx.x  - REC_XY_CENTER)
                 *recon_step;
            Temp = 0.0;
            for(i=0;i<num_belta;i++){
                t = y*cos(i*belta_s) + x*sin(i*belta_s);
                s = x*cos(i*belta_s) - y*sin(i*belta_s);
                so_s = SO/(SO-s);
                t = so_s*t;
                z = so_s*z_o;
                so_s = so_s*so_s;
                z = -z/sample_interval_y + 1.0*Z_CENTER;
                z_int = floor(z);
                z = z-(float)z_int;
                t = t/sample_interval_y + Y_CENTER;
                t_int = floor(t);
                t = t - (float)t_int;

                temp= temp + so_s*((1.0-z)*(1.0-t)
                      *d_miu[i*ZP + z_int*ZP*num_belta + t_int]
                      +z*(1.0-t)*d_miu[i*ZP+(z_int+1)*num_belta*ZP+t_int]
                      +(1.0-z)*t*d_miu[i*ZP+z_int*num_belta*ZP+t_int+1]
                      +z*t*d_miu[i*ZP+(z_int+1)*num_belta*ZP+t_int+1]);
            }
            temp = -0.5*temp*belta_s*sample_interval_y;
            d_ct[id] = (short int)((temp-water)/water*1000.0);
        }
    }
```